class07 lab

Abraham Rachlin

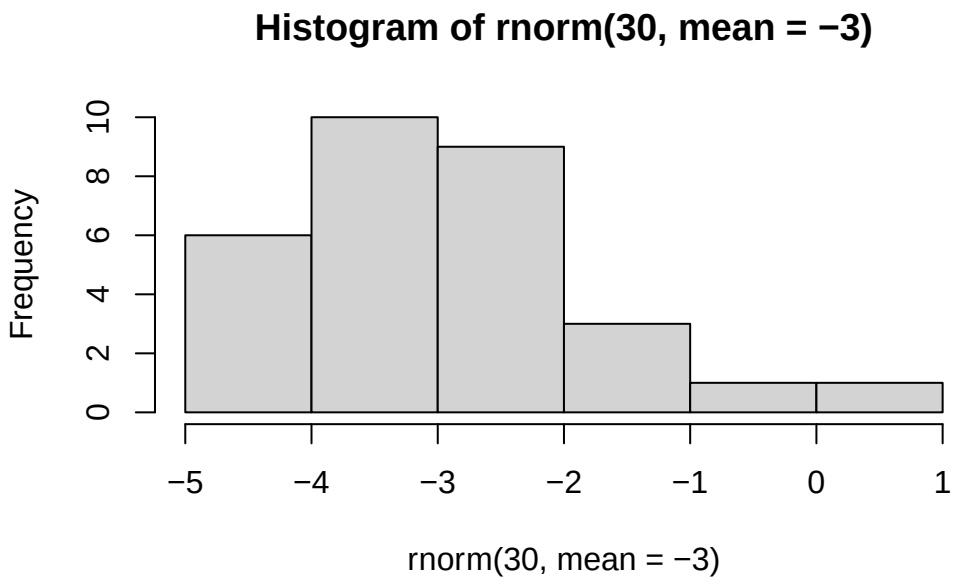
Table of contents

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(30, mean=-3) )
```



Return 30 numbers centered on -3

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

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

x
```

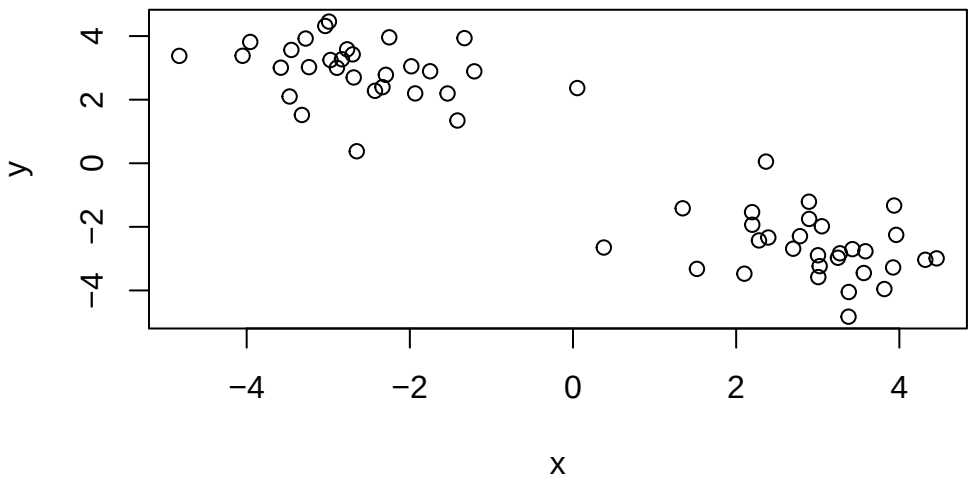
```
                x           y
 [1,] -2.68799145  2.69847350
 [2,] -1.20990245  2.89186669
 [3,] -2.42717552  2.27934265
 [4,] -3.58116315  3.00673868
 [5,] -3.23622969  3.02324678
 [6,] -1.41665658  1.34471775
 [7,] -2.99160865  4.45699216
 [8,]  0.05217801  2.36516980
 [9,] -2.25226971  3.96138272
[10,] -3.32290782  1.51911065
[11,] -2.29411643  2.78183033
[12,] -1.93460301  2.19702350
[13,] -4.82595577  3.37694926
```

```
[14,] -2.70077307  3.42476902
[15,] -2.89375492  3.00283828
[16,] -2.33518980  2.39374251
[17,] -1.53805878  2.19513424
[18,] -4.05002603  3.38038252
[19,] -3.03638322  4.31819162
[20,] -1.33019130  3.93592564
[21,] -3.45242024  3.56429347
[22,] -2.65053382  0.37766886
[23,] -3.27694621  3.92359738
[24,] -2.97252744  3.24766340
[25,] -2.76820904  3.58380778
[26,] -1.75125168  2.89412544
[27,] -2.83197853  3.27175235
[28,] -3.95620924  3.81619851
[29,] -3.47380825  2.10044232
[30,] -1.98184514  3.05013521
[31,]  3.05013521 -1.98184514
[32,]  2.10044232 -3.47380825
[33,]  3.81619851 -3.95620924
[34,]  3.27175235 -2.83197853
[35,]  2.89412544 -1.75125168
[36,]  3.58380778 -2.76820904
[37,]  3.24766340 -2.97252744
[38,]  3.92359738 -3.27694621
[39,]  0.37766886 -2.65053382
[40,]  3.56429347 -3.45242024
[41,]  3.93592564 -1.33019130
[42,]  4.31819162 -3.03638322
[43,]  3.38038252 -4.05002603
[44,]  2.19513424 -1.53805878
[45,]  2.39374251 -2.33518980
[46,]  3.00283828 -2.89375492
[47,]  3.42476902 -2.70077307
[48,]  3.37694926 -4.82595577
[49,]  2.19702350 -1.93460301
[50,]  2.78183033 -2.29411643
[51,]  1.51911065 -3.32290782
[52,]  3.96138272 -2.25226971
[53,]  2.36516980  0.05217801
[54,]  4.45699216 -2.99160865
[55,]  1.34471775 -1.41665658
[56,]  3.02324678 -3.23622969
```

```
[57,]  3.00673868 -3.58116315
[58,]  2.27934265 -2.42717552
[59,]  2.89186669 -1.20990245
[60,]  2.69847350 -2.68799145
```

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, centers=2)
km
```

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

Cluster means:
          x         y
1 -2.637617  2.946117
2  2.946117 -2.637617
```

```
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 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] 52.05104 52.05104
 (between_SS / total_SS =  90.0 %)

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] 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
```

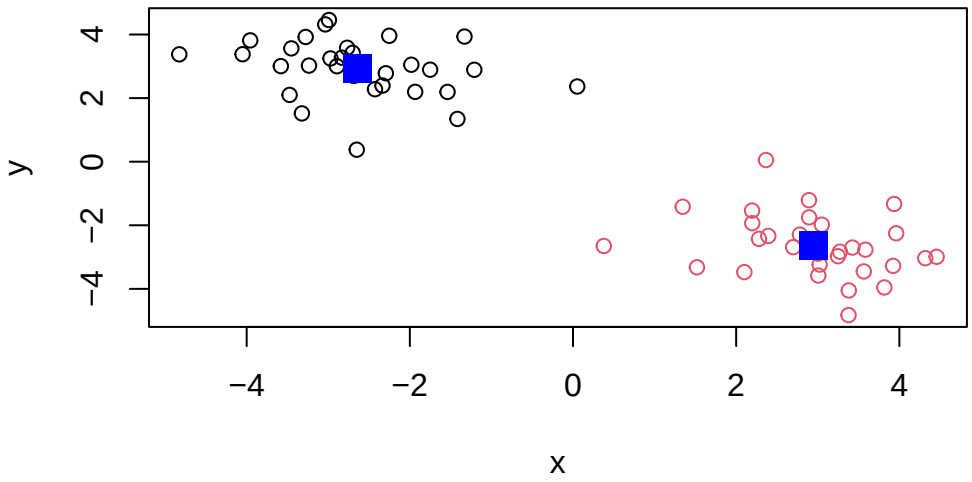
Q. Cluster centers?

```
km$centers
```

```
          x         y
1 -2.637617  2.946117
2  2.946117 -2.637617
```

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

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



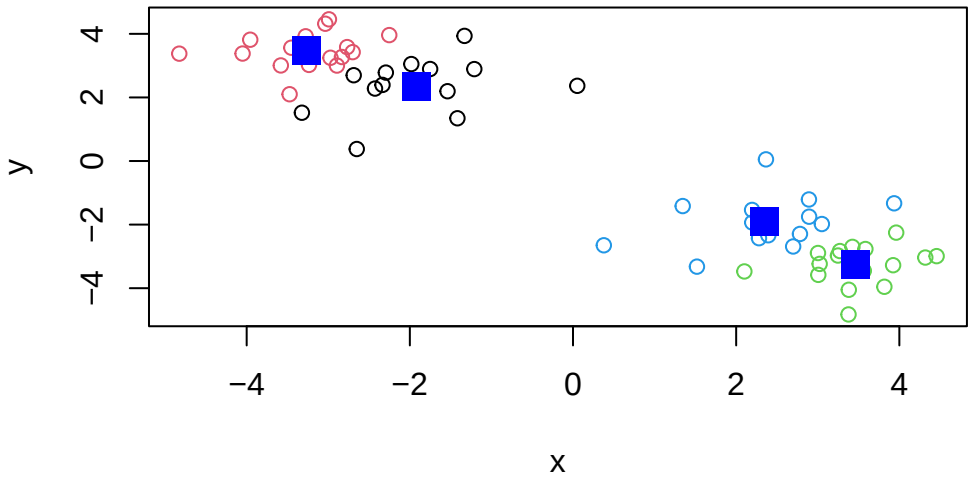
Q. Run `kmeans` again on `x` and this cluster into 4 groups/clusters and plot the same result figure as above.

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

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

Cluster means:
```

```
          x         y
1 -1.916303  2.351733
2 -3.268766  3.466203
3  3.466203 -3.268766
4  2.351733 -1.916303

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

Within cluster sum of squares by cluster:
[1] 18.38286 10.73656 10.73656 18.38286
 (between_SS / total_SS =  94.4 %)

Available components:

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

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



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

limitation is that it can impose a clustering pattern 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()`.

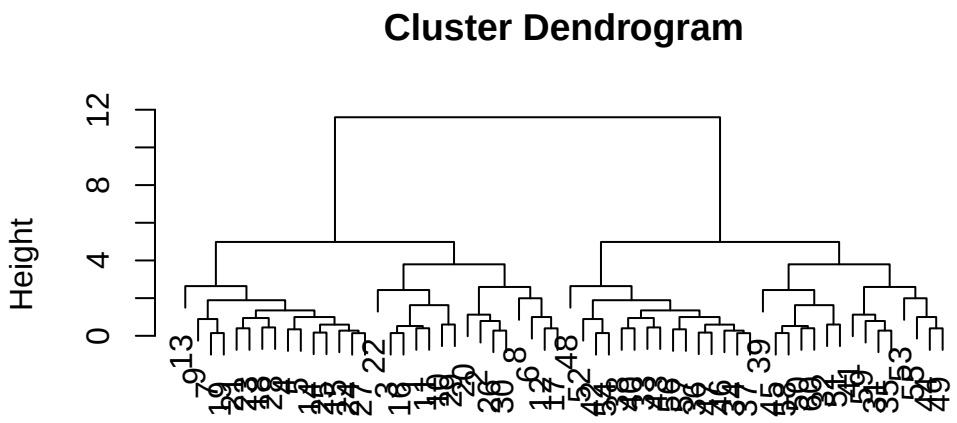
You can't just pass our dataset as is into `hclust()`. You must give "distance matrix" as input. We can get this from the `dist()` function in R.

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

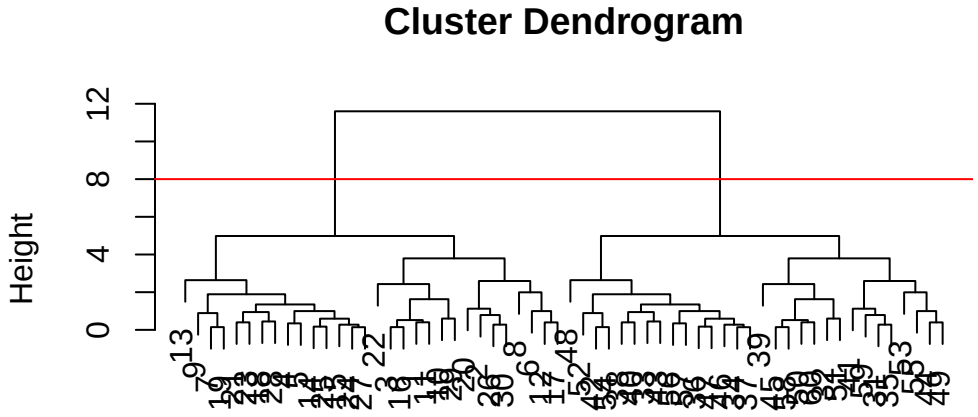
```
Call:
hclust(d = d)

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

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

```
plot(hc)
```




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



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

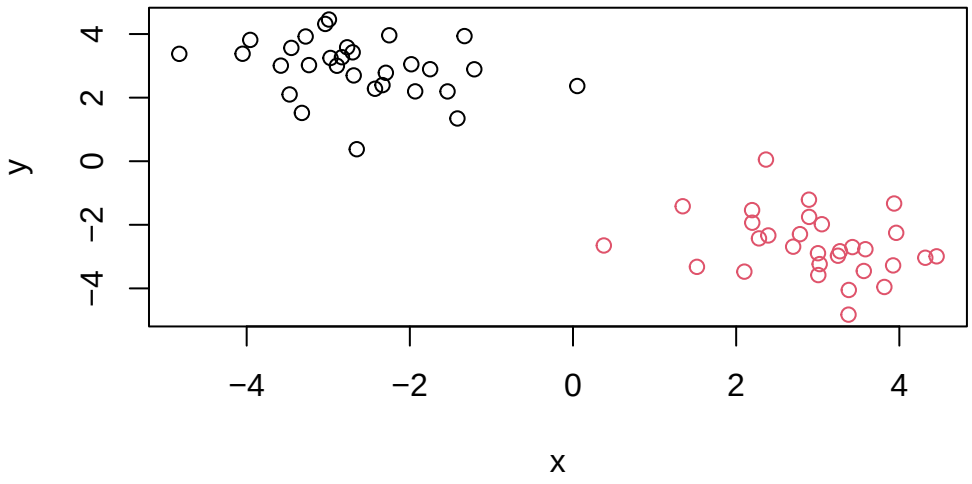
```
grps <- cutree(hc, h=8)
grps
```

```
 [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 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
```

```
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)
```

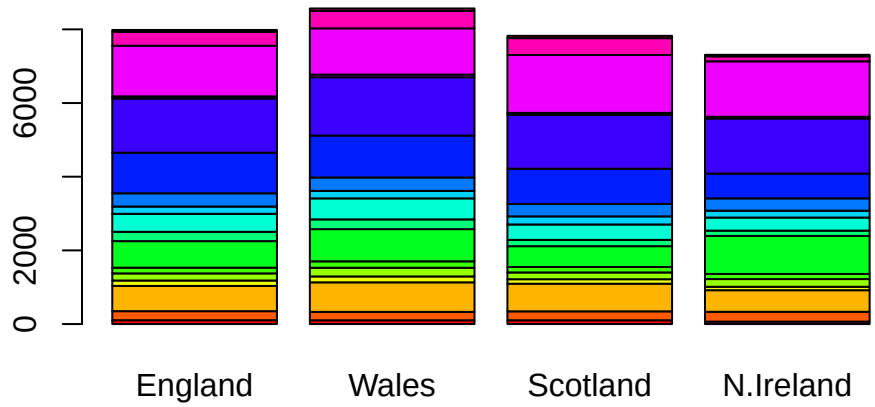
```
              England Wales Scotland N.Ireland
Cheese            105   103      103        66
Carcass_meat      245   227      242       267
Other_meat        685   803      750       586
Fish              147   160      122        93
Fats_and_oils     193   235      184       209
Sugars        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

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

```
              England Wales Scotland N.Ireland
Cheese            105   103      103        66
```

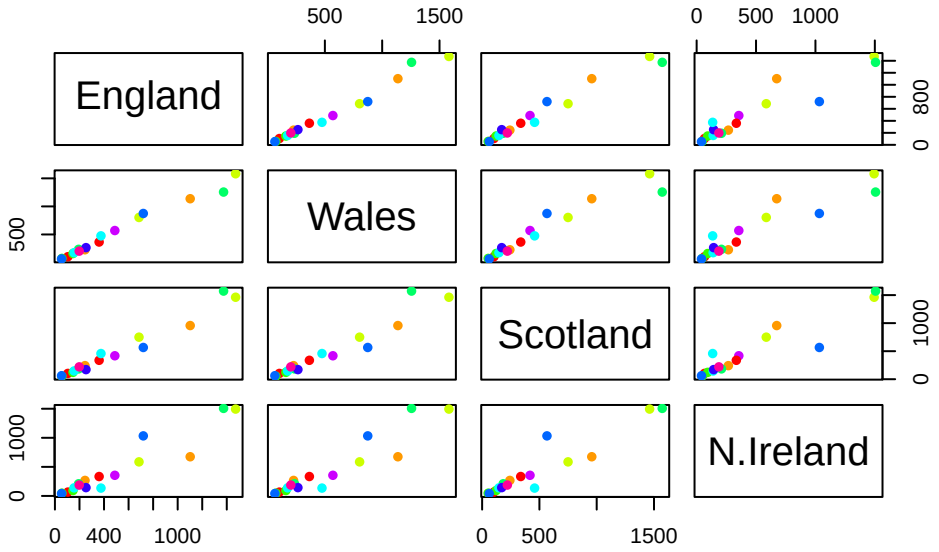
```
Carcass_meat      245   227      242       267
Other_meat        685   803      750       586
Fish              147   160      122        93
Fats_and_oils     193   235      184       209
Sugars            156   175      147       139
```

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



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

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



PCA to the rescue

The main function in base R for PCA is `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 results with five attributes/components.

```
attributes(pca)
```

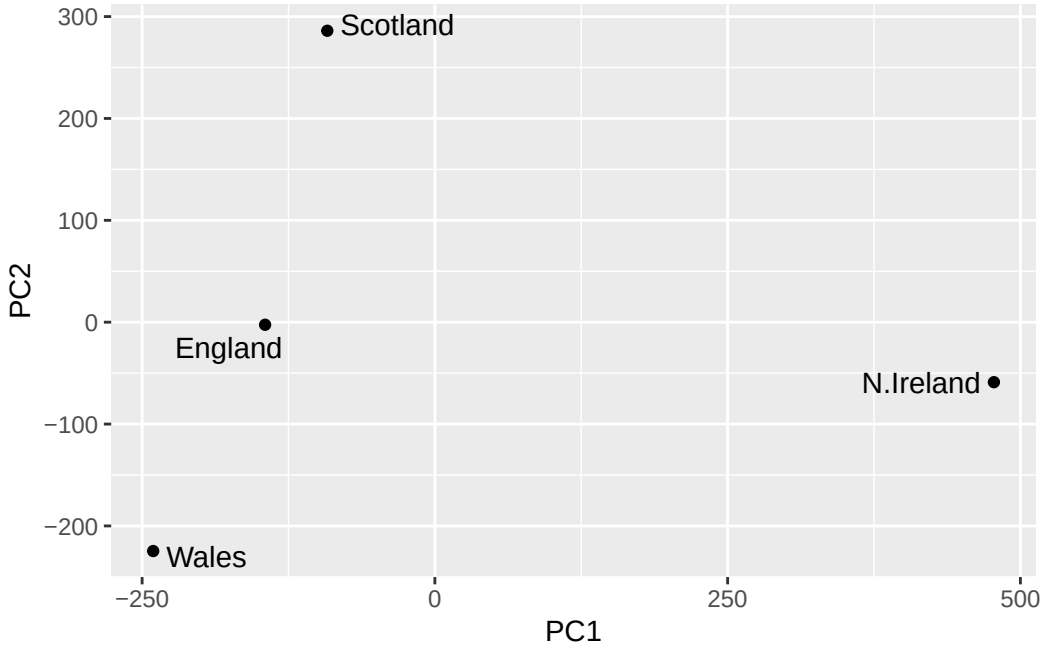
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

The two main results in here are `pca$x` and `pca$rotation`. The first of these (`pca$x`) contains and scores of the data in 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()
```

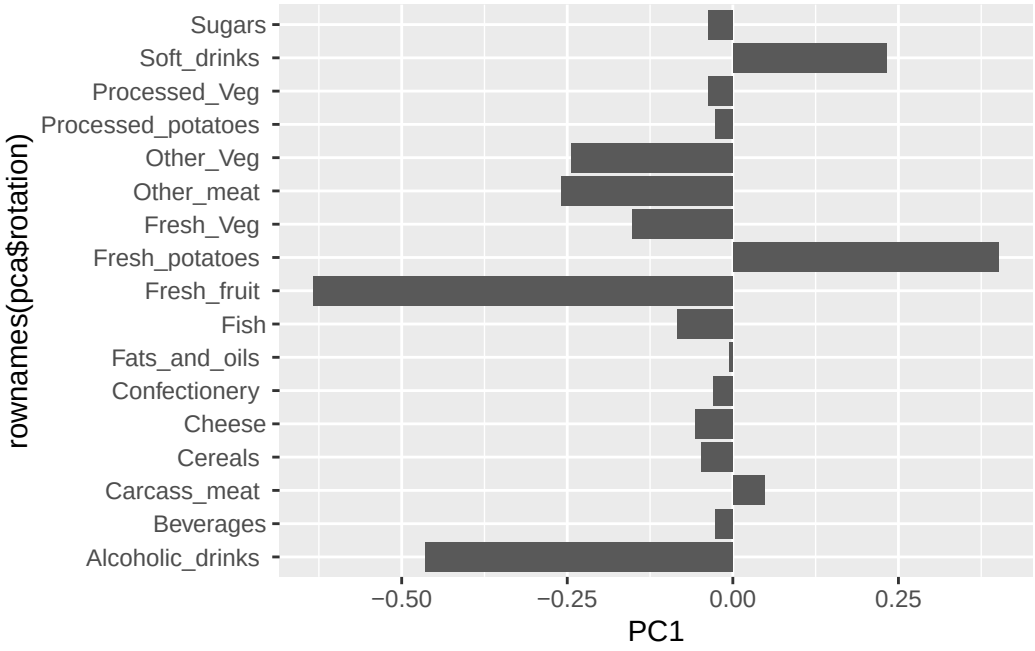


This plot appears to be showing the relationship between PC1 and PC2 when comparing it to all four countries in the U.K. Northern Ireland appears to be the major outlier in this data set. The second major result is contained in the `pca$roration` object or component. Let's plot this to see what PCA is picking up…

```
pca$rotation
```

```
                           PC1          PC2         PC3          PC4
Cheese              -0.056955380  0.016012850  0.02394295 -0.409382587
Carcass_meat         0.047927628  0.013915823  0.06367111  0.729481922
Other_meat          -0.258916658 -0.015331138 -0.55384854  0.331001134
Fish                -0.084414983 -0.050754947  0.03906481  0.022375878
Fats_and_oils       -0.005193623 -0.095388656 -0.12522257  0.034512161
Sugars              -0.037620983 -0.043021699 -0.03605745  0.024943337
Fresh_potatoes       0.401402060 -0.715017078 -0.20668248  0.021396007
Fresh_Veg           -0.151849942 -0.144900268  0.21382237  0.001606882
Other_Veg           -0.243593729 -0.225450923 -0.05332841  0.031153231
Processed_potatoes  -0.026886233  0.042850761 -0.07364902 -0.017379680
Processed_Veg       -0.036488269 -0.045451802  0.05289191  0.021250980
Fresh_fruit         -0.632640898 -0.177740743  0.40012865  0.227657348
Cereals             -0.047702858 -0.212599678 -0.35884921  0.100043319
Beverages           -0.026187756 -0.030560542 -0.04135860 -0.018382072
Soft_drinks          0.232244140  0.555124311 -0.16942648  0.222319484
Alcoholic_drinks    -0.463968168  0.113536523 -0.49858320 -0.273126013
Confectionery       -0.029650201  0.005949921 -0.05232164  0.001890737
```

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



This plot is showing the relationship between PC1 and the 17 different dimensions that are found in the data set. It appears that fruit and alcoholic drinks appear to be negative outliers, while potatoes and soft drinks appear to be positive outliers.