Class 7: Machine Learning 1

Canbin Cai (A18087473)

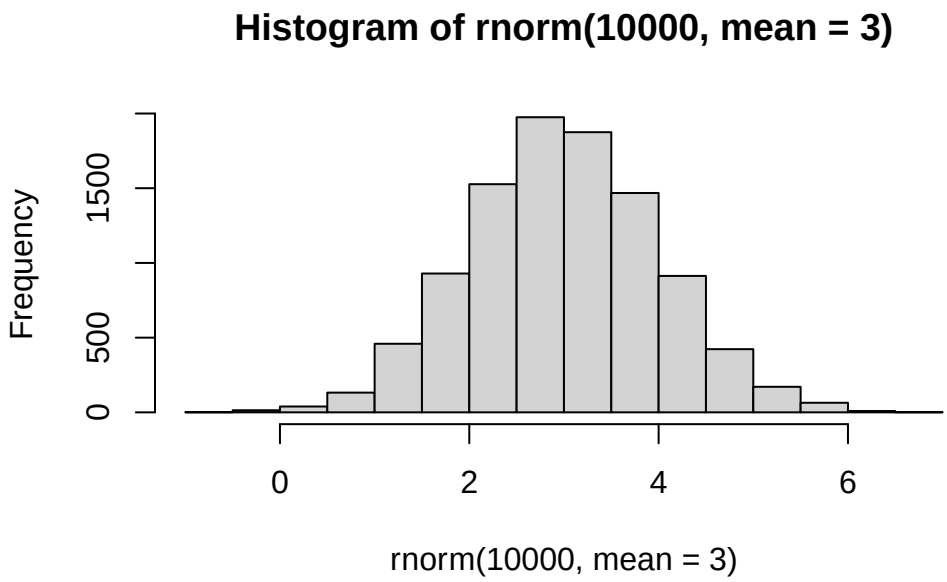
Table of contents

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
```

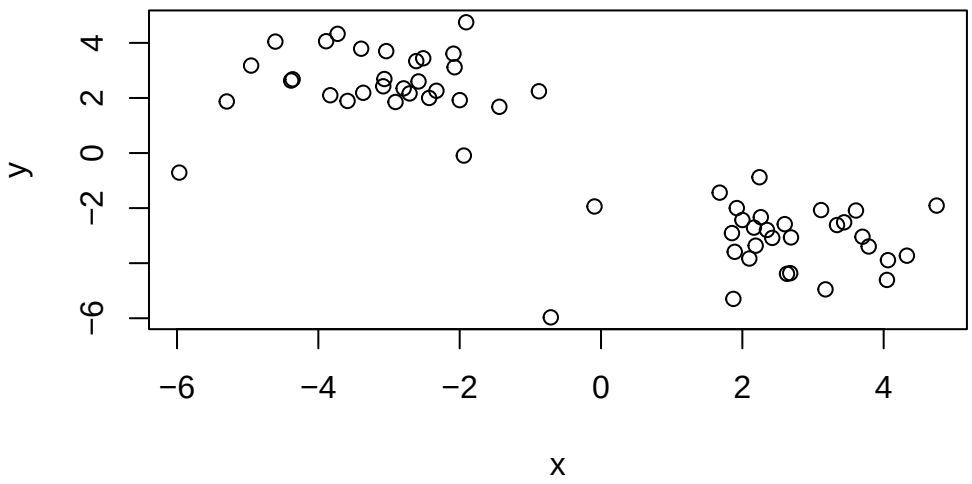
```
                x           y
 [1,] -3.03976425  3.69972524
 [2,] -2.58227048  2.59969992
 [3,] -4.38510970  2.63190633
 [4,] -3.39416568  3.78829008
 [5,] -4.95129483  3.17636435
 [6,] -2.07195271  3.11405293
 [7,] -3.89031810  4.06045109
 [8,] -1.99890091  1.92141889
 [9,] -1.90814351  4.74927061
[10,] -3.83142730  2.09865036
[11,] -3.58681877  1.89229904
[12,] -1.94128341 -0.09186285
[13,] -5.29619365  1.87265161
```

```
[14,] -2.79301618  2.34814040
[15,] -2.43194209  2.00078784
[16,] -3.36500043  2.18837701
[17,] -4.60820742  4.04639441
[18,] -3.72794633  4.32847880
[19,] -2.32955295  2.26133152
[20,] -2.90798166  1.85290517
[21,] -5.96801911 -0.71053360
[22,] -3.06558351  2.68837716
[23,] -2.08837903  3.60789032
[24,] -3.08138657  2.42323176
[25,] -2.51585577  3.44050406
[26,] -4.36371818  2.67737811
[27,] -1.43839833  1.67919506
[28,] -0.87832873  2.24225892
[29,] -2.61462726  3.33932963
[30,] -2.70922242  2.16406111
[31,]  2.16406111 -2.70922242
[32,]  3.33932963 -2.61462726
[33,]  2.24225892 -0.87832873
[34,]  1.67919506 -1.43839833
[35,]  2.67737811 -4.36371818
[36,]  3.44050406 -2.51585577
[37,]  2.42323176 -3.08138657
[38,]  3.60789032 -2.08837903
[39,]  2.68837716 -3.06558351
[40,] -0.71053360 -5.96801911
[41,]  1.85290517 -2.90798166
[42,]  2.26133152 -2.32955295
[43,]  4.32847880 -3.72794633
[44,]  4.04639441 -4.60820742
[45,]  2.18837701 -3.36500043
[46,]  2.00078784 -2.43194209
[47,]  2.34814040 -2.79301618
[48,]  1.87265161 -5.29619365
[49,] -0.09186285 -1.94128341
[50,]  1.89229904 -3.58681877
[51,]  2.09865036 -3.83142730
[52,]  4.74927061 -1.90814351
[53,]  1.92141889 -1.99890091
[54,]  4.06045109 -3.89031810
[55,]  3.11405293 -2.07195271
[56,]  3.17636435 -4.95129483
```

```
[57,]  3.78829008 -3.39416568
[58,]  2.63190633 -4.38510970
[59,]  2.59969992 -2.58227048
[60,]  3.69972524 -3.03976425
```

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  2.603034 -3.125494
```

```
2 -3.125494  2.603034

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] 79.59639 79.59639
 (between_SS / total_SS =  86.1 %)

Available components:

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

The `kmeans()` function return 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
```

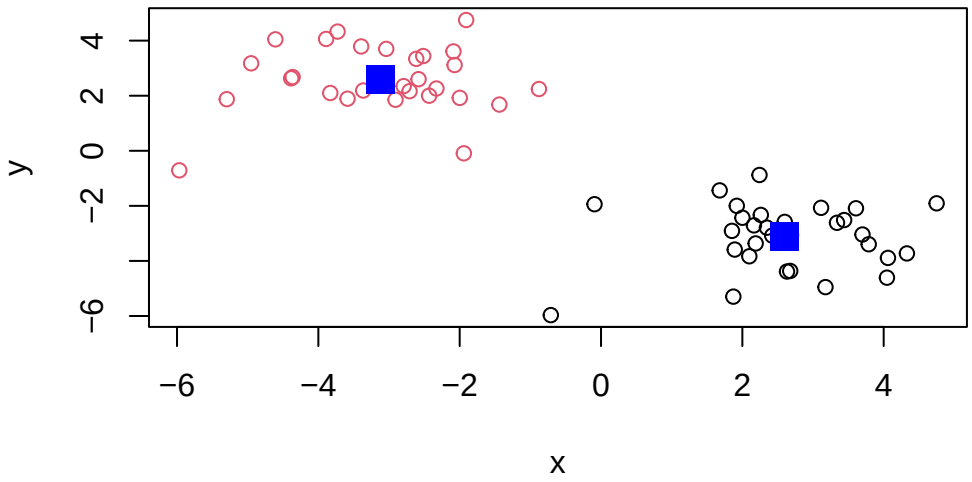
Q. Cluster centers?

```
km$centers
```

```
          x         y
1  2.603034 -3.125494
2 -3.125494  2.603034
```

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)
```



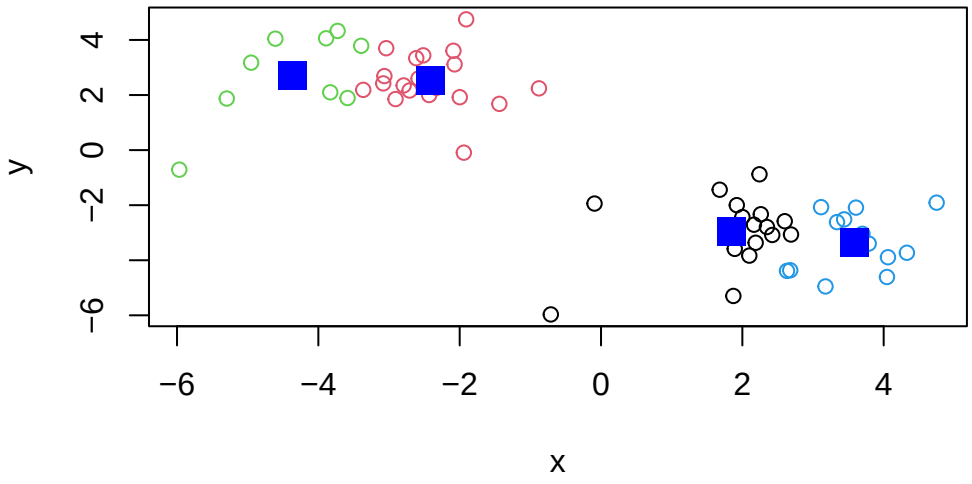
Q. Run `kmeans()` again on `x` and this cluster into 4 groups/clusters and plot the same result figure as above.

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

```
K-means clustering with 4 clusters of sizes 17, 19, 11, 13

Cluster means:
```

```
          x         y
1  1.848882 -2.953255
2 -2.408505  2.538352
3 -4.363929  2.714757
4  3.589234 -3.350729

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

Within cluster sum of squares by cluster:
[1] 38.19614 25.37739 27.36387 17.92407
 (between_SS / total_SS =  90.5 %)

Available components:

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

```r
plot(x, col=km4$cluster)
points(km4$centers, col="blue", pch=15, cex=2)
```



Key-point: K-means clustering is super popular but can be miss-used. 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 hierarchincal clustering is called `hclust()`.

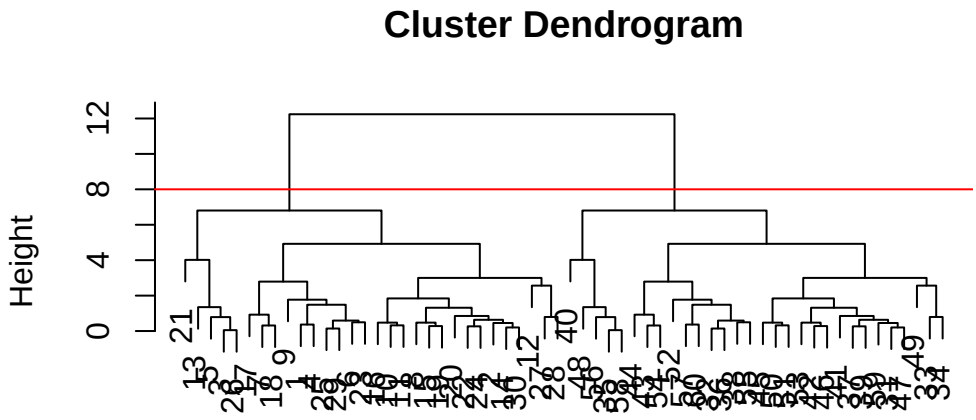
You can't just pass our dataset as is into `hclust()` you must give "distance matrix" as input. We can get this from the `dist()` function in R.

```
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...

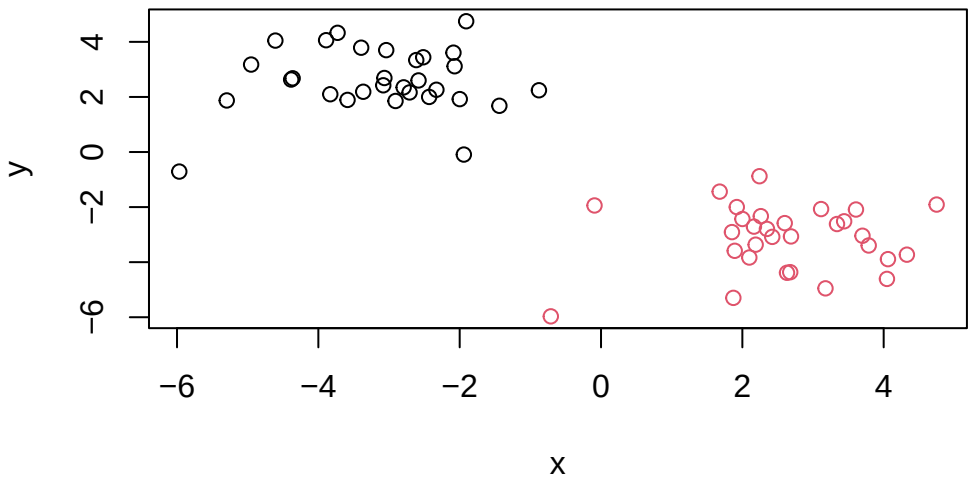
```
grps <- cutree(hc, h=8)
grps
```

```
 [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 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 - particulary 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
```

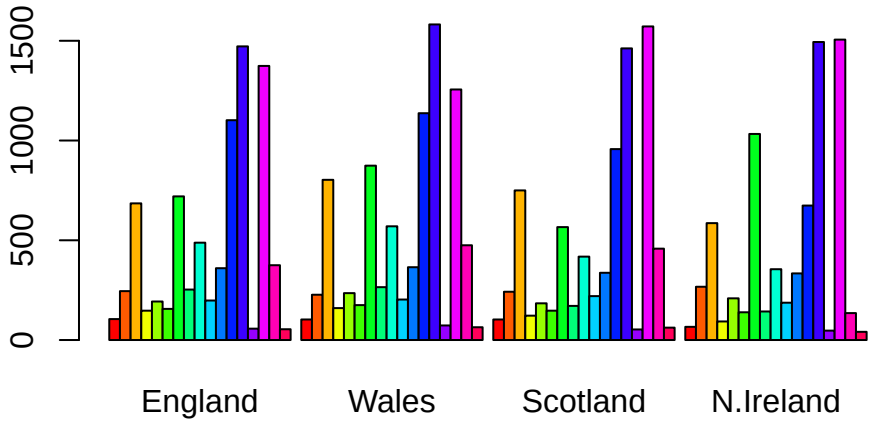
```
rownames(x) <- x[,1]
x <- x[,-1]
head(x)
```

```
               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
```

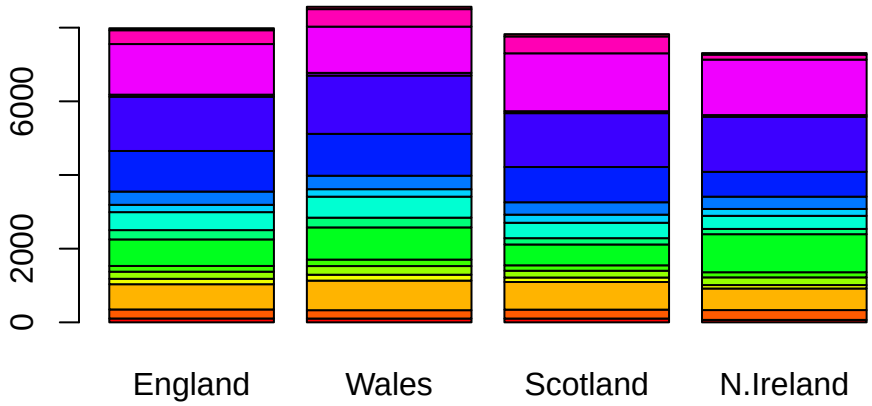
```
x <- read.csv(url, row.names = 1)
head(x)
```

```
               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=T, col=rainbow(nrow(x)))
```

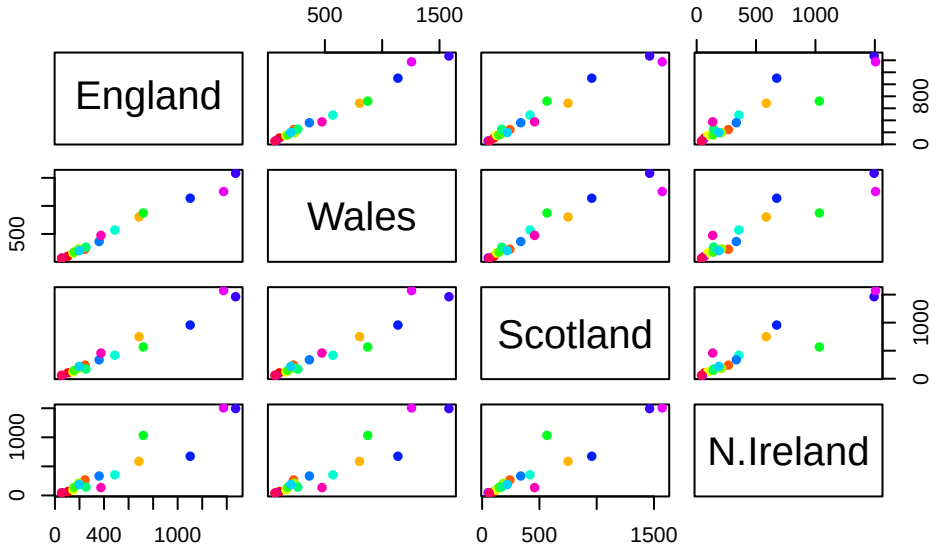



```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



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

```
pairs(x, col=rainbow(nrow(x)), 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"
```

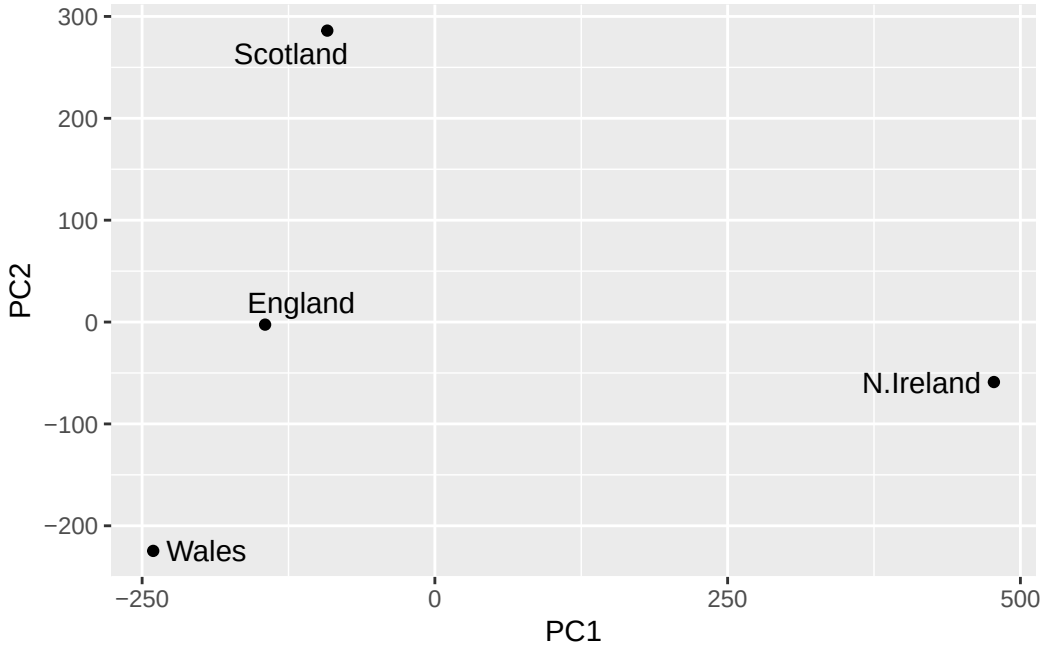
The two main "results" in here are `pca$x` and `pca$rotation`. The first of these (`pca$x`) contains the scores of the data on the new PC axis - we use these to make our "PCA plot".

```
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
```

Interperting PCA Results

```
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()
```

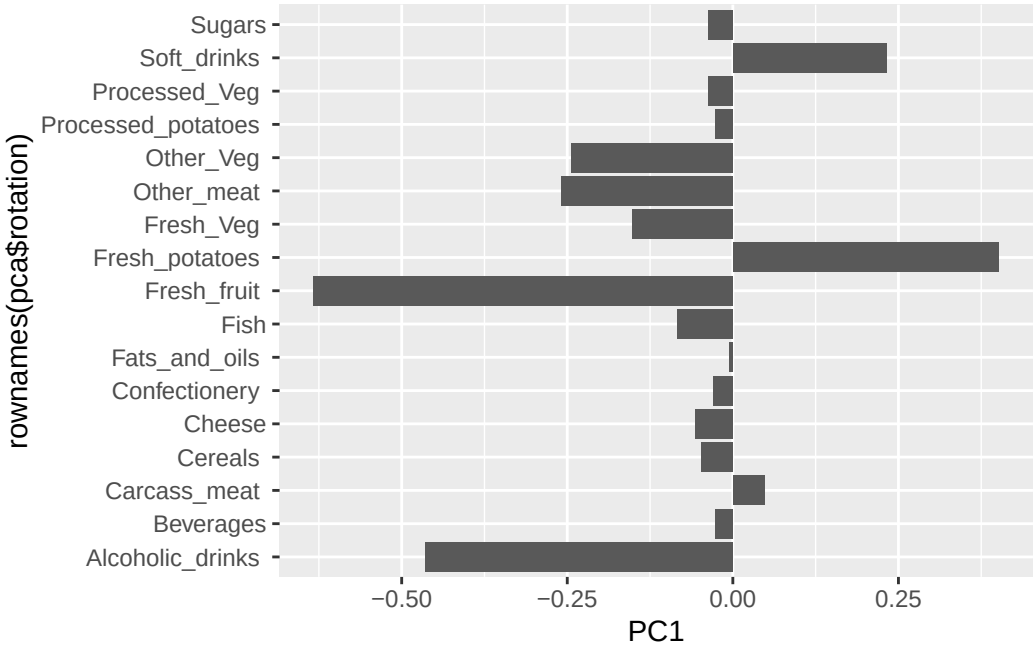


This plot shows the differences between the two PCAs, where the distance between each point shows the variable for the different countries, i.e. the average consumption of the 17 food types.

N. Ireland is more far away towards the PC1, so their average eating habits may be quite different from the other three countries. Similarly, Scotland is more far away towards PC2, and they may also have different eating habits. England and Wales are pretty close, where their diet may be influenced by both food variables.

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 plot shows the food variables in PC1, with each bar reflecting the consumption of each of the 17 food items.

In connection with the first plot (PC1 vs. PC2), N. Ireland is on the far right, so they are likely to have consumed a lot more fresh potatoes and soft drinks compared to the other three countries. Meanwhile, Scotland, England, and Wales are on the left side of the plot, so they are likely to consume more fresh fruit and alcoholic drinks.