Class07

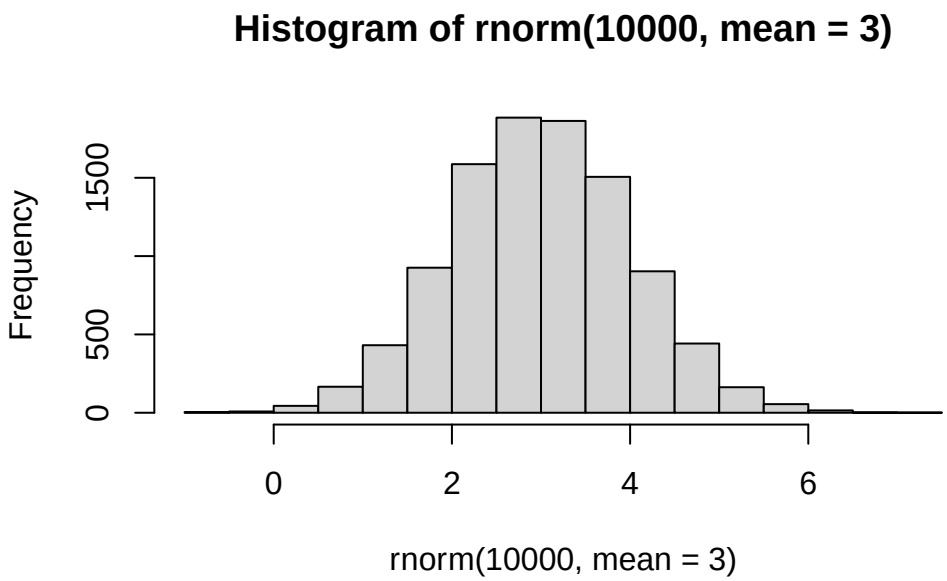
Jihyun In

Table of contents

Today we will explore unsupervised machine learning methods starting iwht clustering and idmensionality reduction.

Clustering

```
hist(rnorm(10000, mean=3))
```



Teturn 30 numbers centred on -3.

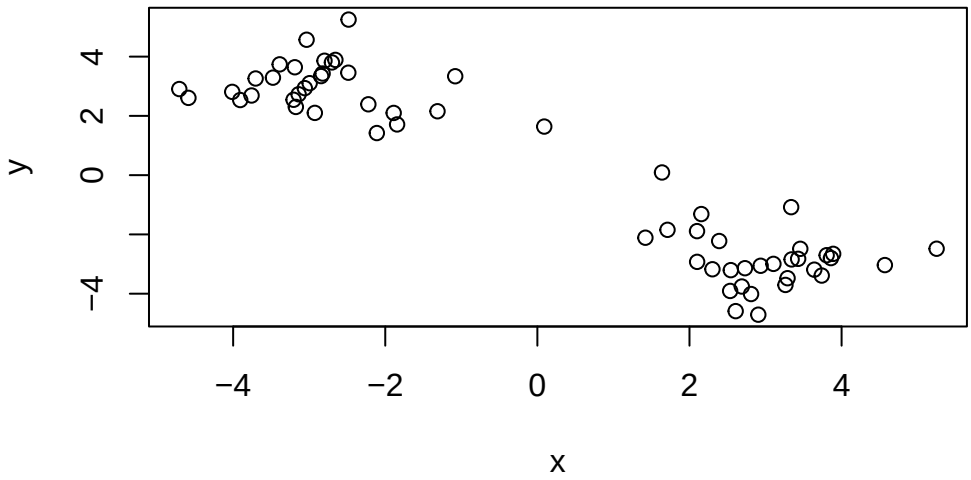
```
tmp <- c(rnorm(30, mean=-3),
rnorm(30, mean=3))

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

x
```

```
                x           y
 [1,] -3.20546058  2.54354935
 [2,] -4.01210719  2.80987311
 [3,] -2.48576559  3.45704869
 [4,] -3.70442220  3.26191390
 [5,] -2.82381683  3.42881455
 [6,] -2.99421845  3.10369991
 [7,] -1.88918414  2.09923572
 [8,] -2.22236255  2.39069781
 [9,] -3.13942717  2.72939533
[10,] -3.75833060  2.68806846
[11,] -2.92599248  2.10060332
[12,] -3.03446702  4.56908274
[13,] -2.48254400  5.24980391
[14,] -2.79905002  3.86025997
[15,] -3.38842381  3.73903959
[16,] -3.90703649  2.53485270
[17,] -3.05781169  2.93575739
[18,] -3.17592803  2.30193143
[19,] -2.70122642  3.80363814
[20,] -4.58820314  2.60742450
[21,] -4.70844970  2.90518464
[22,] -1.08041707  3.33707343
[23,] -2.84397600  3.34314029
[24,]  0.09040073  1.63896629
[25,] -3.18939770  3.64095010
[26,] -2.11077281  1.42028065
[27,] -3.47700387  3.28966439
[28,] -1.31314883  2.15583176
[29,] -1.84442821  1.71115679
[30,] -2.65585681  3.88924748
[31,]  3.88924748 -2.65585681
[32,]  1.71115679 -1.84442821
```

```
[33,]  2.15583176 -1.31314883
[34,]  3.28966439 -3.47700387
[35,]  1.42028065 -2.11077281
[36,]  3.64095010 -3.18939770
[37,]  1.63896629  0.09040073
[38,]  3.34314029 -2.84397600
[39,]  3.33707343 -1.08041707
[40,]  2.90518464 -4.70844970
[41,]  2.60742450 -4.58820314
[42,]  3.80363814 -2.70122642
[43,]  2.30193143 -3.17592803
[44,]  2.93575739 -3.05781169
[45,]  2.53485270 -3.90703649
[46,]  3.73903959 -3.38842381
[47,]  3.86025997 -2.79905002
[48,]  5.24980391 -2.48254400
[49,]  4.56908274 -3.03446702
[50,]  2.10060332 -2.92599248
[51,]  2.68806846 -3.75833060
[52,]  2.72939533 -3.13942717
[53,]  2.39069781 -2.22236255
[54,]  2.09923572 -1.88918414
[55,]  3.10369991 -2.99421845
[56,]  3.42881455 -2.82381683
[57,]  3.26191390 -3.70442220
[58,]  3.45704869 -2.48576559
[59,]  2.80987311 -4.01210719
[60,]  2.54354935 -3.20546058
```

Make a plot o fx

```
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  2.984873 -2.847628
2 -2.847628  2.984873

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] 50.49808 50.49808
 (between_SS / total_SS =  91.0 %)

Available components:

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

The `kmeans()` function returns a "list" with 9 ccomponents. 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
```

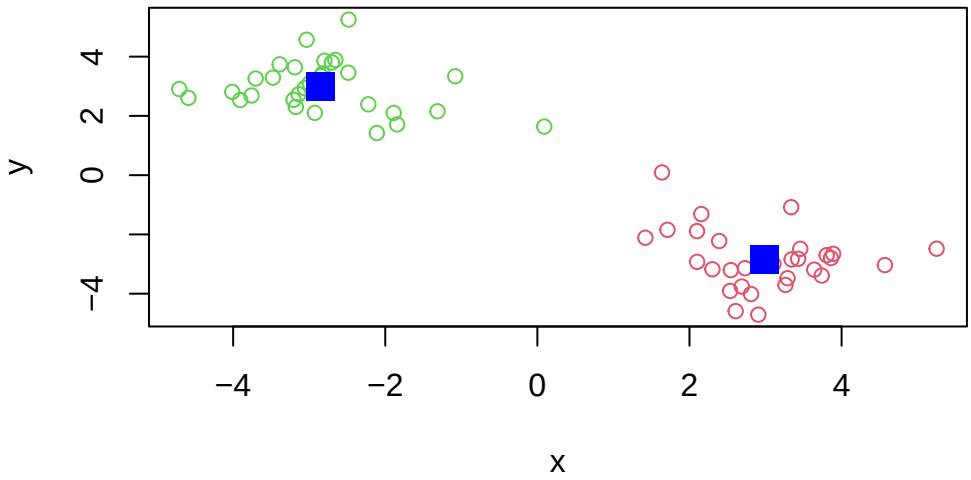
Q. Cluster centers?

```
km$centers
```

```
          x         y
1  2.984873 -2.847628
2 -2.847628  2.984873
```

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

```
plot(x, col=km$cluster + 1) #didn't like black as a point color
points(km$centers, col="blue", pch=15, cex=2)
```

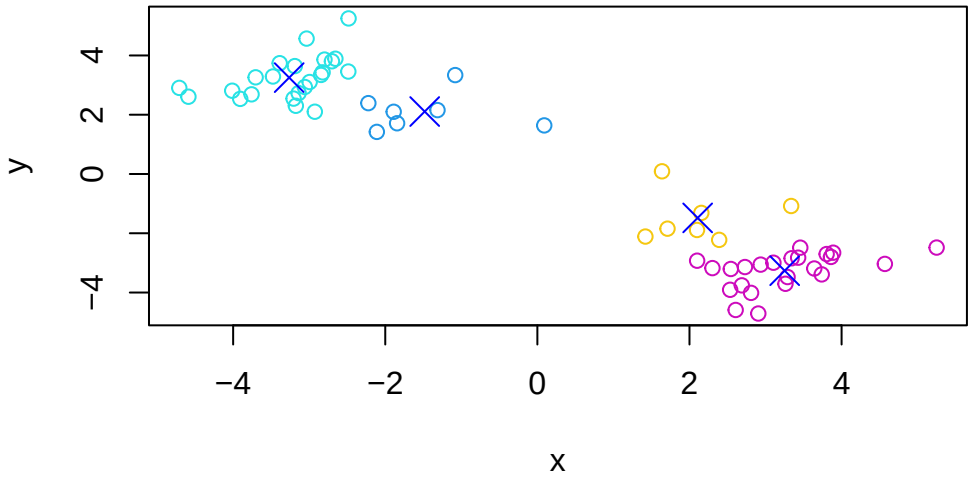


Q. Run `kmeans()` again on `x` and this cluster into 4 groups/clusters and plot the same result figure as above.

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

```
K-means clustering with 4 clusters of sizes 7, 23, 23, 7

Cluster means:
          x         y
1 -1.481416  2.107606
2 -3.263431  3.251867
3  3.251867 -3.263431
4  2.107606 -1.481416

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

Within cluster sum of squares by cluster:
[1]  6.346205 20.082856 20.082856  6.346205
 (between_SS / total_SS =  95.3 %)

Available components:

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

```
plot(x, col=km4$cluster + 3) #didn't like black as a point color
points(km4$centers, col="blue", pch=4, cex=2) #changed pch shape for preference
```



Key-point: K-means clustering is super popular but cna be misused. One big limitation is that it can impose a clustering pattern on your data even if clear natrual grouping doesn't exist - i.e. it does what you tell it to do in terms of `centers`.

Hierarchial clustering

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

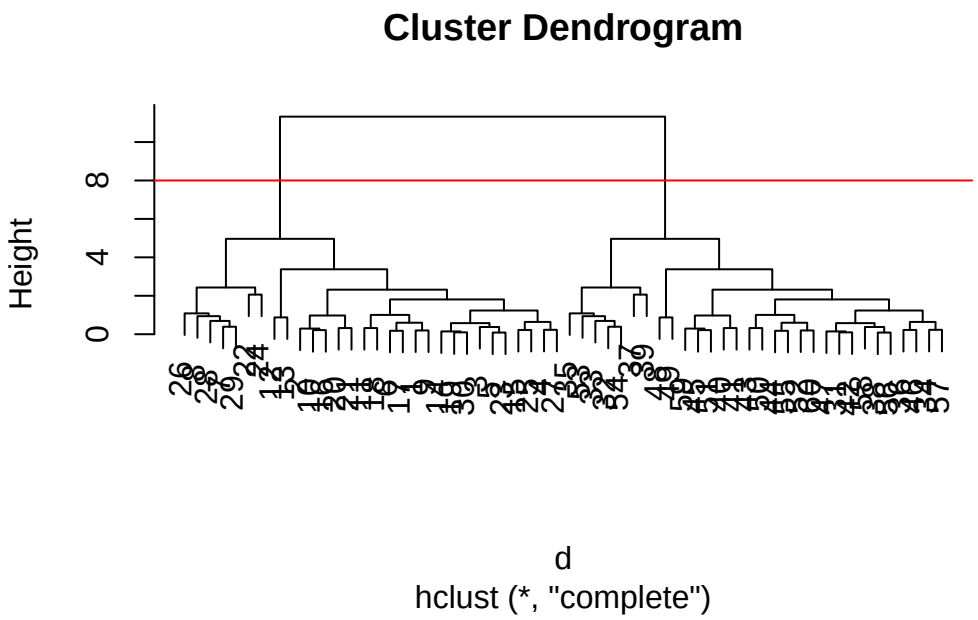
You can't just pass our dataset as is into `hclust()`, you must give "distance matrix" as input. We can get this from the `dist()` function in R.

```
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()` method but do have a special `plot()` method.

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



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

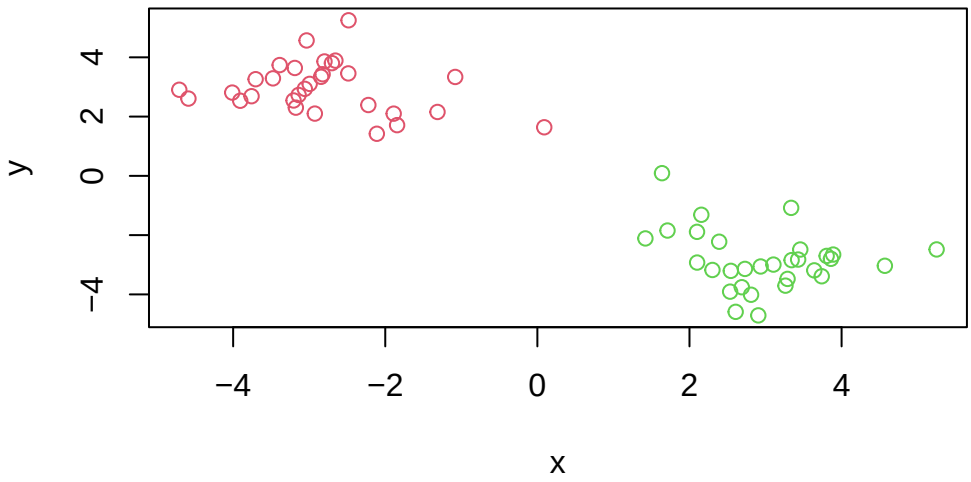
```
grps <- cutree(hc, h=8)
grps
```

```
 [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 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(grps)
```

```
grps
 1  2
30 30
```

```
plot(x, col=grps +1) #Again, I like colors
```



Hierarchial Clustering is distinct in that the dendrogram(tree figure) can reveal the potential grouping in your data (unlike K-means).

Principal Component Analysis

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

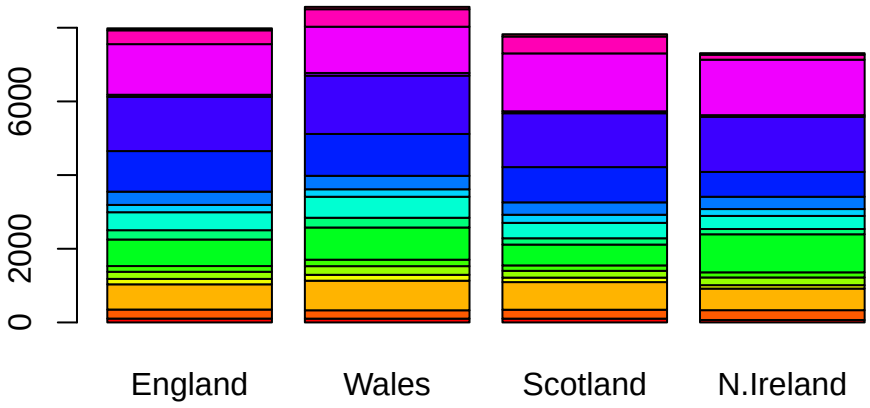
```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
head(x)
```

```
              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
```

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

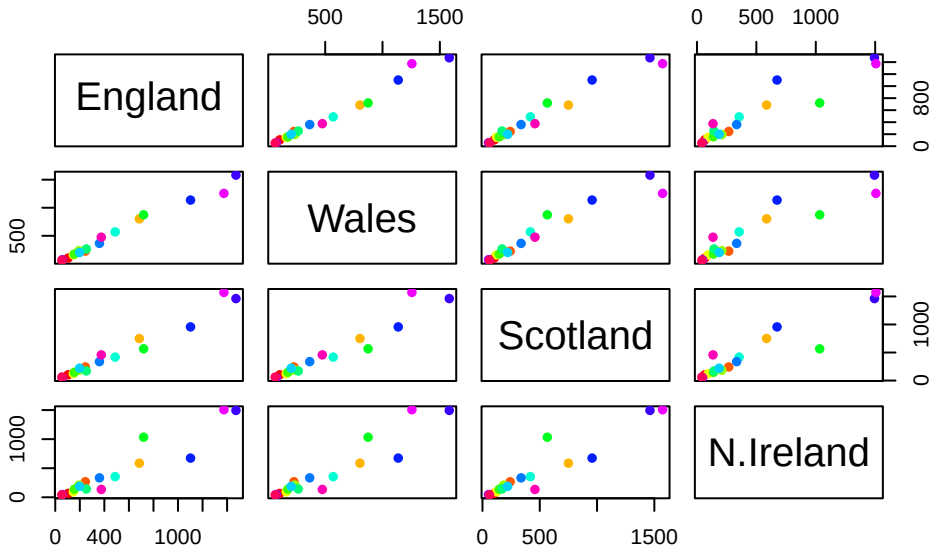
```
              England Wales Scotland N.Ireland
Cheese            105   103      103        66
Carcass_meat      245   227      242       267
Other_meat        685   803      750       586
Fish              147   160      122        93
Fats_and_oils     193   235      184       209
Sugars        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

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



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

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



PCA to the rescue

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

```
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 <-prcomp(t(x))
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
```

The `precomp()` function returns a list object of our results with

```
attributes(pca)
```

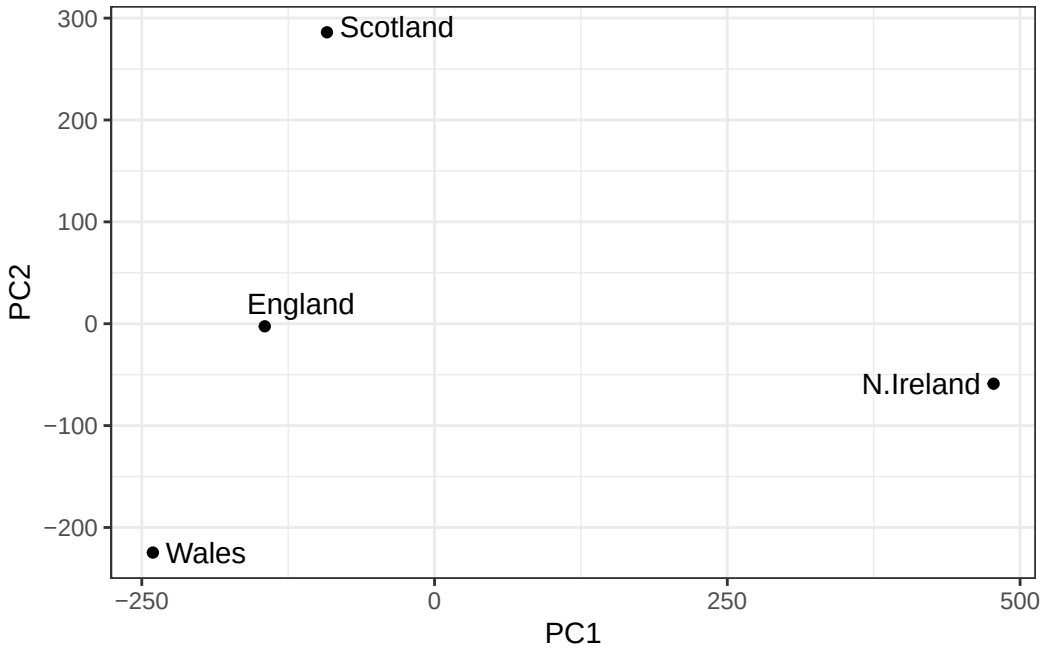
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

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

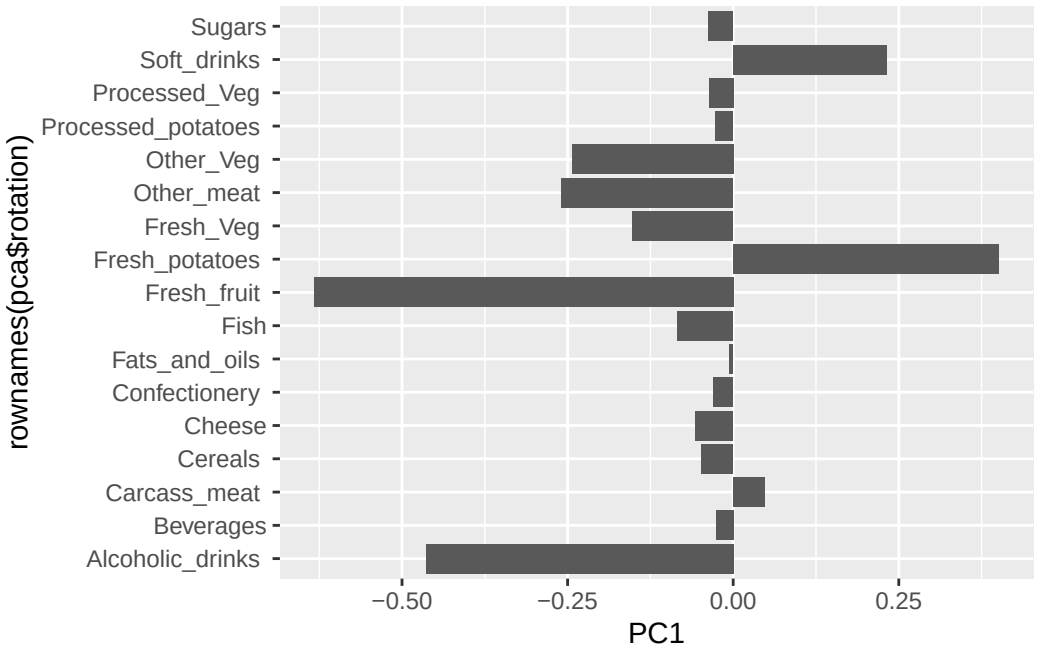
```
library(ggplot2)
library(ggrepel)
#Make  aplot of pca$x with PC1 vs. PC2

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



the above plot plots PC2 vs. PC1, showing the scores along each axis. This shows that along PC1, N.Ireland is an outlier, and along PC2, Scotland is far from wales.

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()
```



The above bar plot shows how much each category of food contributes to PC1. Fresh potatoes and soft drinks explain most of the positive variance (what Ireland, the positive outlier, consumes more of), and fresh fruit and alcoholic drinks explain a large part of what Ireland consumes less of.