Class 7: Machine Learning 1

Linh Dang (A16897764)

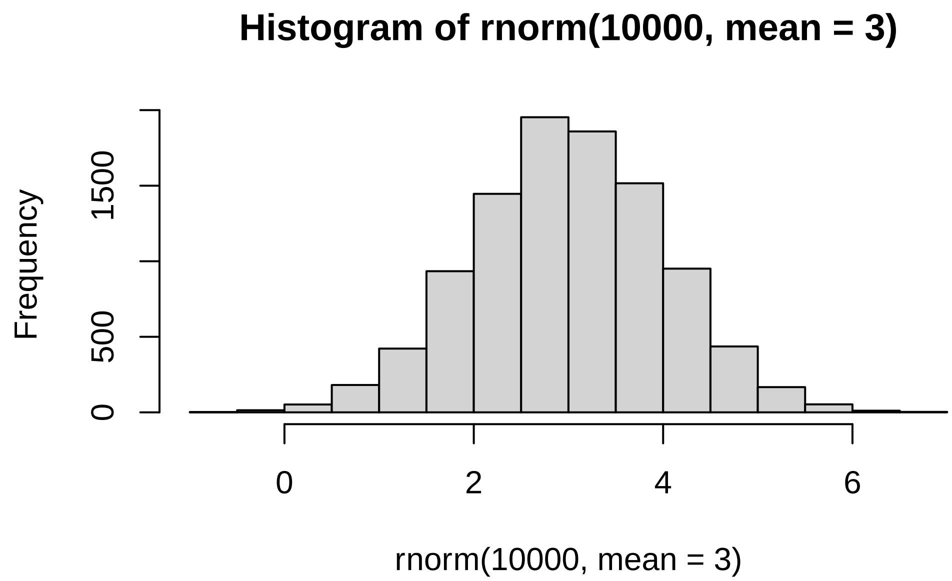
Table of contents

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 up 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
```

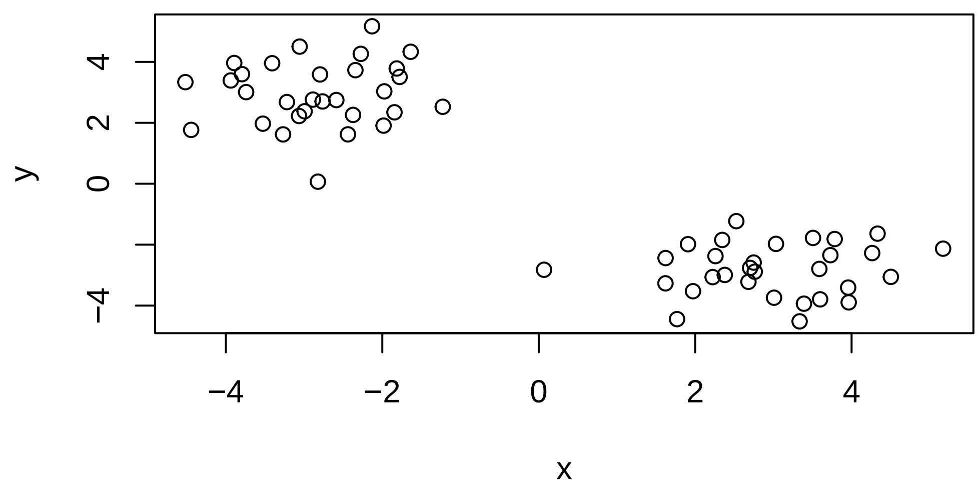
```
                x          y
 [1,] -1.84452539  2.34596990
 [2,] -3.06459895  2.22358175
 [3,] -3.74116542  3.00818175
 [4,] -3.52720387  1.97220865
 [5,] -3.05687199  4.50164059
 [6,] -2.88711317  2.76191692
 [7,] -1.77886014  3.50648277
 [8,] -1.81551868  3.78380299
 [9,] -2.99377354  2.37851007
[10,] -3.21986415  2.68104517
[11,] -3.89312473  3.96319889
[12,] -1.98385371  1.90814731
[13,] -1.97457417  3.03306000
```

```
[14,] -2.34386515  3.72880775
[15,] -2.58760534  2.74768761
[16,] -2.79626851  3.58765830
[17,] -1.22743831  2.52605143
[18,] -1.63741341  4.33208126
[19,] -2.37416380  2.25941685
[20,] -4.51774654  3.33553013
[21,] -2.27527840  4.26314963
[22,] -2.13100872  5.16989774
[23,] -2.76350327  2.70213124
[24,] -2.43943024  1.62161988
[25,] -3.40844078  3.95604778
[26,] -2.82358985  0.06789587
[27,] -4.44358043  1.76827808
[28,] -3.79421710  3.59766222
[29,] -3.26880967  1.62012933
[30,] -3.93737538  3.39070358
[31,]  3.39070358 -3.93737538
[32,]  1.62012933 -3.26880967
[33,]  3.59766222 -3.79421710
[34,]  1.76827808 -4.44358043
[35,]  0.06789587 -2.82358985
[36,]  3.95604778 -3.40844078
[37,]  1.62161988 -2.43943024
[38,]  2.70213124 -2.76350327
[39,]  5.16989774 -2.13100872
[40,]  4.26314963 -2.27527840
[41,]  3.33553013 -4.51774654
[42,]  2.25941685 -2.37416380
[43,]  4.33208126 -1.63741341
[44,]  2.52605143 -1.22743831
[45,]  3.58765830 -2.79626851
[46,]  2.74768761 -2.58760534
[47,]  3.72880775 -2.34386515
[48,]  3.03306000 -1.97457417
[49,]  1.90814731 -1.98385371
[50,]  3.96319889 -3.89312473
[51,]  2.68104517 -3.21986415
[52,]  2.37851007 -2.99377354
[53,]  3.78380299 -1.81551868
[54,]  3.50648277 -1.77886014
[55,]  2.76191692 -2.88711317
[56,]  4.50164059 -3.05687199
```

```
[57,]  1.97220865 -3.52720387
[58,]  3.00818175 -3.74116542
[59,]  2.22358175 -3.06459895
[60,]  2.34596990 -1.84452539
```

Make a plot of `x`

```r
plot(x)
```



K-means

The main function in "base" R for K-means clustering is called `kmeans`:

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

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

Cluster means:
          x         y
1  2.958083 -2.818359
```

```
2 -2.818359  2.958083

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 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] 54.02804 54.02804
 (between_SS / total_SS =  90.3 %)

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.

```r
attributes(km)
```

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

$class
[1] "kmeans"
```

> Q. How mnay points are in each cluster?

```r
km$size
```

```
[1] 30 30
```

> Q. Cluster assignment/membership vector?

```r
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 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
```

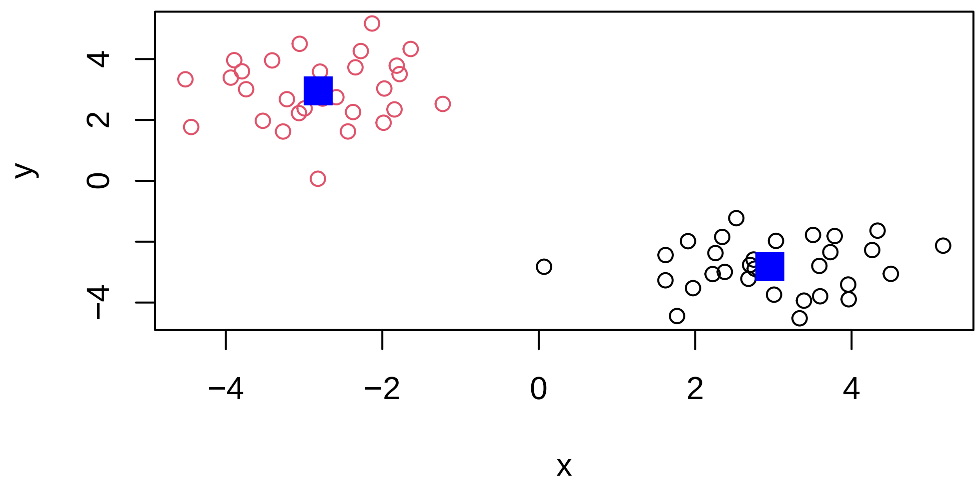
> Q. Cluster centers?

```
km$centers
```

```
          x         y
1  2.958083 -2.818359
2 -2.818359  2.958083
```

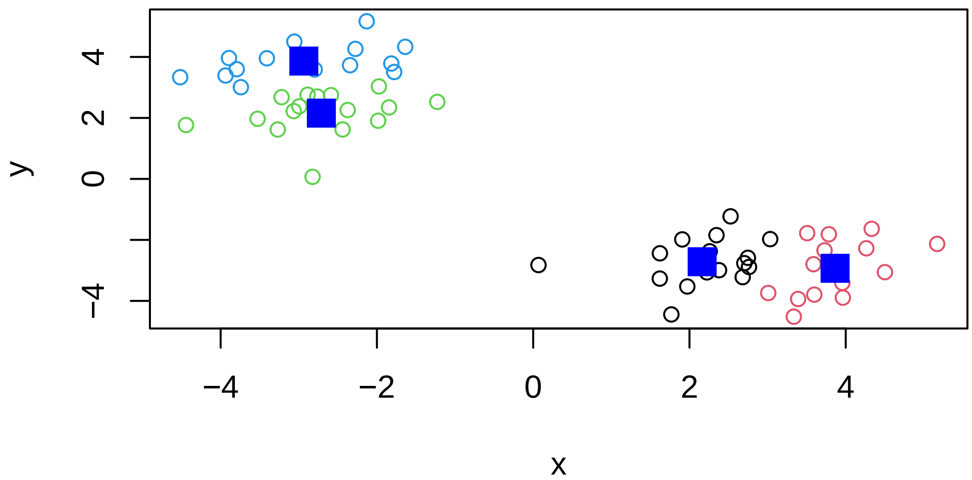
Q. Make a plot of our `kmeans()` results showing cluster assignment using different colors for each cluster/group of points and cluster centers.

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



Q. Run `kmeans()` again on `x` and this cluster in to 4 groups/clusters and plot the same result 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 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 Hierarchical Clustering is called `hclust()`.

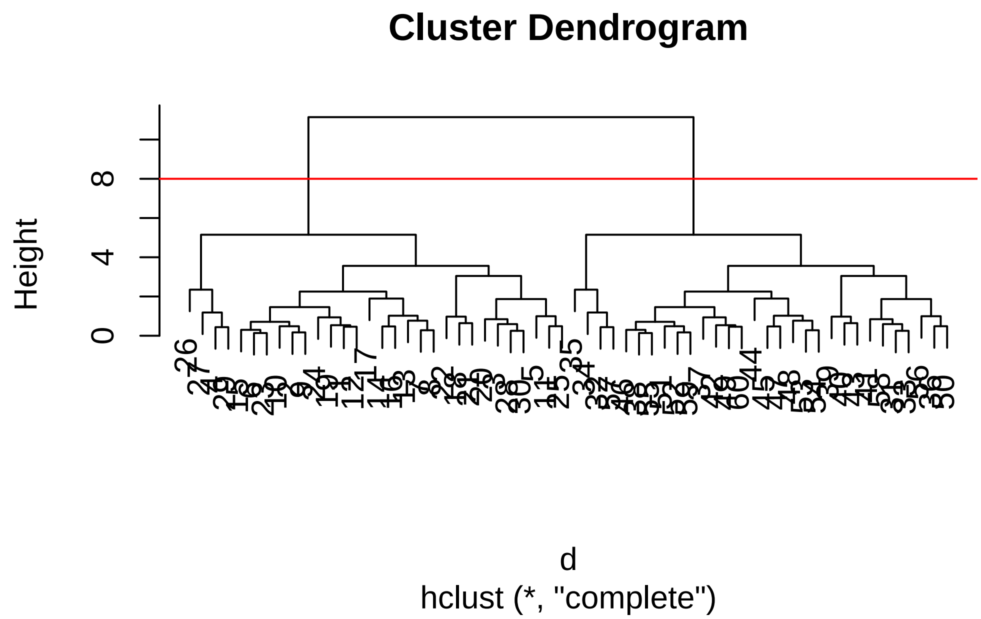
You can't just pass our dataset as is into `hclust()` you must give "distance matrix" as input. We can get this from the `dist()` function in R.

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

```
Call:
hclust(d = d)

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

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

```r
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...

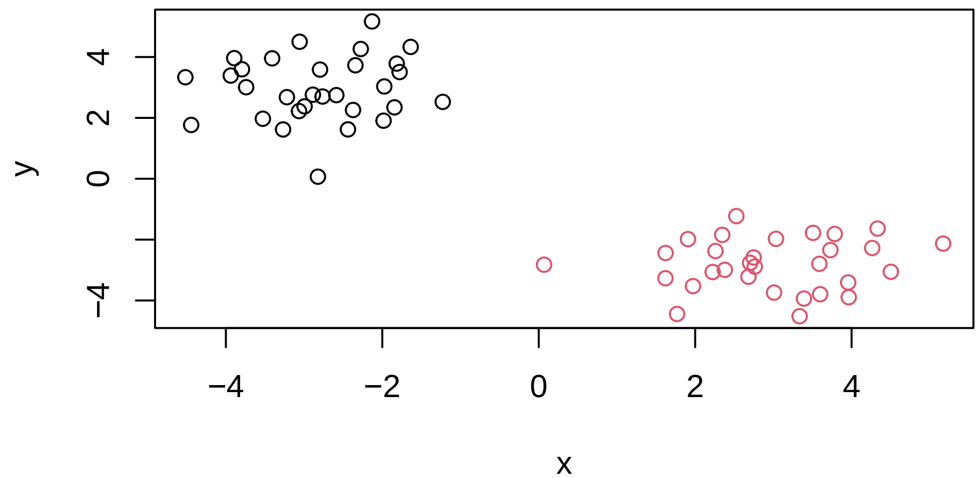
```r
grps <- cutree(hc, h=8)
grps
```

```
 [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
```

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



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

Principal Compoenent Analysis (PCA)

PCA is a common and highly useful dimensionality reduction technique used in many field - particularly bioinformatics.

Here we will analyze some data from the UK on the 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
```

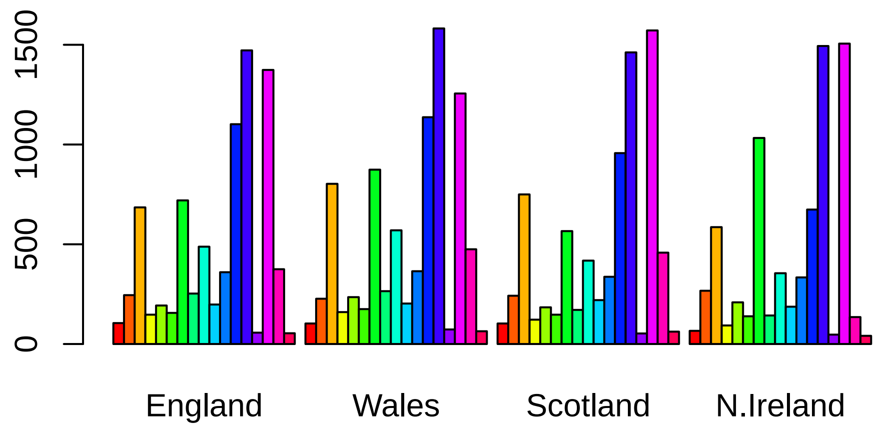
```r
rownames(x) <- x[,1]
x <- x[,-1]
head(x)
```

```
               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
```

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

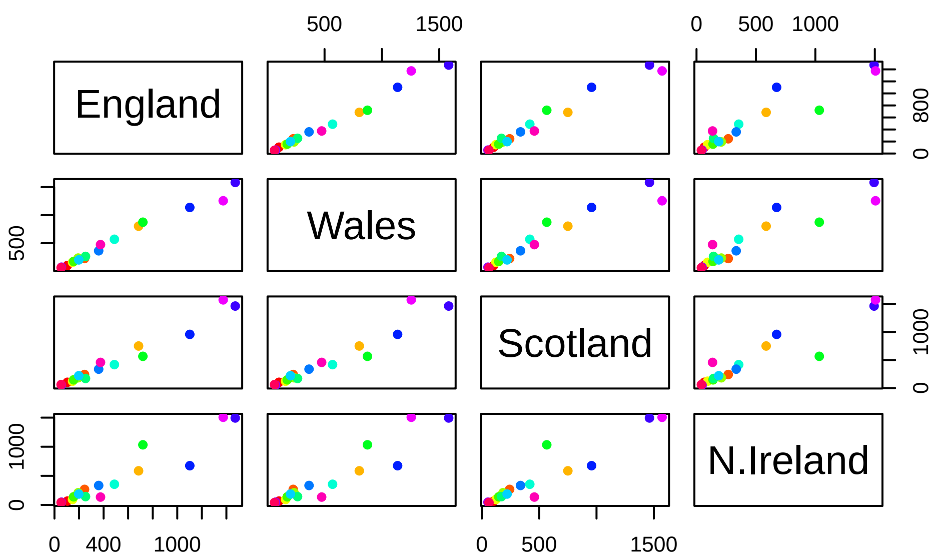
```
               England Wales Scotland N.Ireland
Cheese             105   103      103        66
Carcass_meat       245   227      242       267
Other_meat         685   803      750       586
Fish               147   160      122        93
Fats_and_oils      193   235      184       209
Sugars  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

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



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

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



PCA to the rescue

The main function in base R for PCA `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)
```

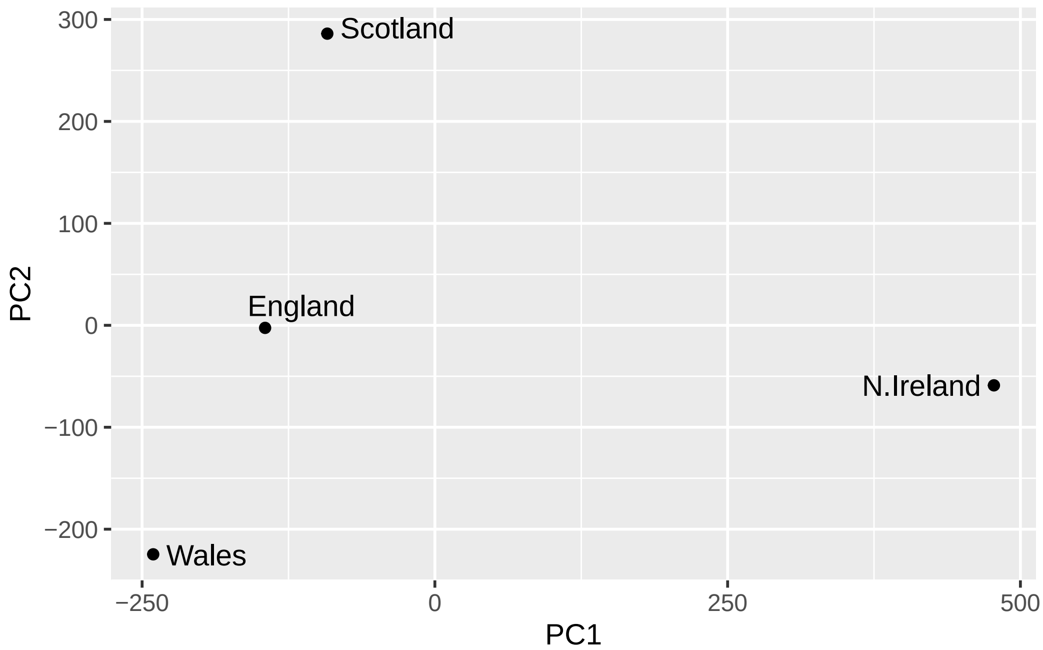
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

The two main "result" 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
```

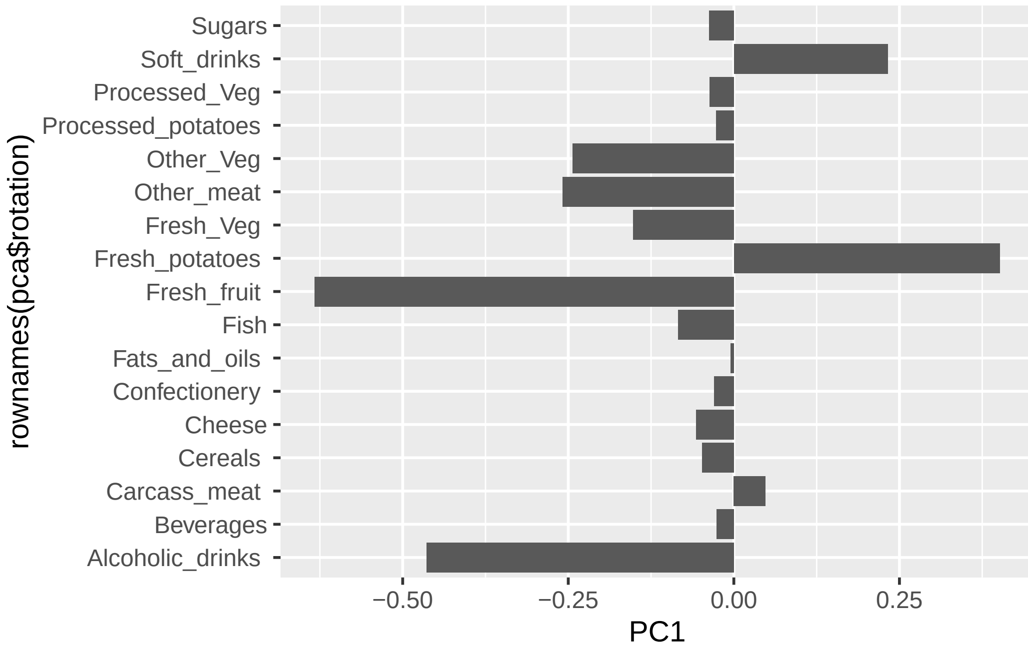
```
library(ggplot2)
library(ggrepel)

# 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 show the variation of food between the 4 countries. In this case, N.Ireland is most dissimilar to the 3 other countries by intepreting the relationship on the PC (N.Ireland has positive score compared to England, Wales, and Scotland).

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 plot shows the contribution of variance going in both negative (England, Wale, and Scotland) and positive direction (N.Ireland). We can see that N.Ireland contribute to the right side of the plot for soft drinks and fresh potatoes, while the other 3 countries contribute to the lest side of the plot for fresh fruit and alcoholic drinks, etc.