Class_07_Clustering_013124

Johann Tailor

Clustering

Today we will learn about machine learning and how to use the function `kmeans()`

Kmeans clustering

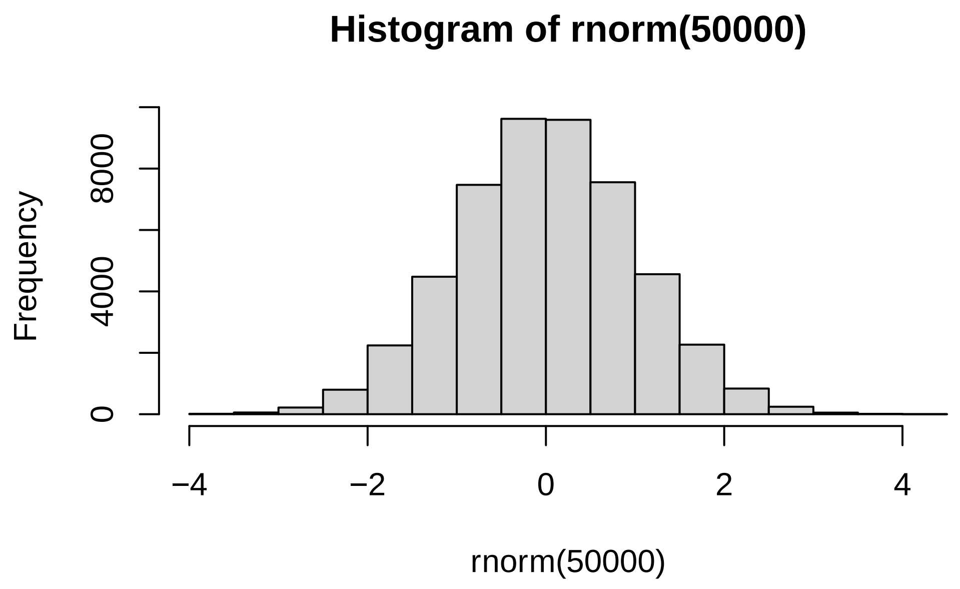
Let's see how it works:

This gives random numbers or n; Its a way to make up random data:

```r
rnorm(10)
```

```
 [1] -0.691717584 -0.551251259  0.484945936  0.033734703 -0.290303723
 [6] -0.633913038  0.148153427  0.009847386 -0.928668178 -0.095386375
```

```r
hist( rnorm(50000))
```



```
# mean will be 0 and sd is 1. This is the deafult.
# check the input available
```

Make a little vector with 60 total points that re centered at +3 and half that are at -3.

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

```
 [1]  3.5243176  3.4235841  2.8067860  2.3380526  2.0256185  3.3659074
 [7]  2.0095389  1.9065568  3.1006812  1.8011001  2.8495788  1.3684813
[13]  3.7254322  1.6350836  3.5249288  2.5443319  2.4515152  4.0967365
[19]  3.3215667  3.5545912  1.4144193  3.5334997  2.7005013  1.7844940
[25]  5.0013754  0.7522507  1.8194114  3.6884536  3.7954554  2.1655766
[31] -2.5745901 -2.9506615 -3.2944968 -5.3727302 -2.6557116 -2.6886282
[37] -4.4613512 -5.6063706 -3.5374885 -3.8207248 -4.3433144 -1.8882028
[43] -2.9780096 -0.9389722 -2.3700056 -2.3563444 -2.9242484 -3.5257087
[49] -0.9477753 -4.0929673 -0.8042800 -3.3573342 -2.8091477 -3.0676778
[55] -2.7997984 -4.2903461 -3.7145639 -1.6572943 -2.9218879 -1.3260166
```

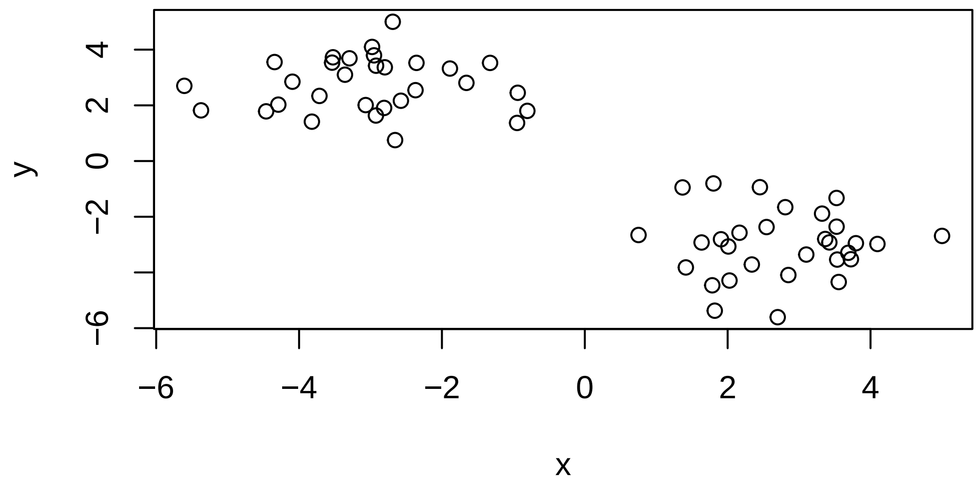
Let’s allign it so that it goes -ve to +ve and make a plot to visually see two different clusters.

```r
rev.tmp <- rev(tmp)


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

```
              x          y
 [1,]  3.5243176 -1.3260166
 [2,]  3.4235841 -2.9218879
 [3,]  2.8067860 -1.6572943
 [4,]  2.3380526 -3.7145639
 [5,]  2.0256185 -4.2903461
 [6,]  3.3659074 -2.7997984
 [7,]  2.0095389 -3.0676778
 [8,]  1.9065568 -2.8091477
 [9,]  3.1006812 -3.3573342
[10,]  1.8011001 -0.8042800
[11,]  2.8495788 -4.0929673
[12,]  1.3684813 -0.9477753
[13,]  3.7254322 -3.5257087
[14,]  1.6350836 -2.9242484
[15,]  3.5249288 -2.3563444
[16,]  2.5443319 -2.3700056
[17,]  2.4515152 -0.9389722
[18,]  4.0967365 -2.9780096
[19,]  3.3215667 -1.8882028
[20,]  3.5545912 -4.3433144
[21,]  1.4144193 -3.8207248
[22,]  3.5334997 -3.5374885
[23,]  2.7005013 -5.6063706
[24,]  1.7844940 -4.4613512
[25,]  5.0013754 -2.6886282
[26,]  0.7522507 -2.6557116
[27,]  1.8194114 -5.3727302
[28,]  3.6884536 -3.2944968
[29,]  3.7954554 -2.9506615
[30,]  2.1655766 -2.5745901
[31,] -2.5745901  2.1655766
[32,] -2.9506615  3.7954554
[33,] -3.2944968  3.6884536
[34,] -5.3727302  1.8194114
[35,] -2.6557116  0.7522507
```

```
[36,] -2.6886282  5.0013754
[37,] -4.4613512  1.7844940
[38,] -5.6063706  2.7005013
[39,] -3.5374885  3.5334997
[40,] -3.8207248  1.4144193
[41,] -4.3433144  3.5545912
[42,] -1.8882028  3.3215667
[43,] -2.9780096  4.0967365
[44,] -0.9389722  2.4515152
[45,] -2.3700056  2.5443319
[46,] -2.3563444  3.5249288
[47,] -2.9242484  1.6350836
[48,] -3.5257087  3.7254322
[49,] -0.9477753  1.3684813
[50,] -4.0929673  2.8495788
[51,] -0.8042800  1.8011001
[52,] -3.3573342  3.1006812
[53,] -2.8091477  1.9065568
[54,] -3.0676778  2.0095389
[55,] -2.7997984  3.3659074
[56,] -4.2903461  2.0256185
[57,] -3.7145639  2.3380526
[58,] -1.6572943  2.8067860
[59,] -2.9218879  3.4235841
[60,] -1.3260166  3.5243176
```

```
plot(x)
```



Let's run the **kmeans()** feature to see how many clusters are there:

```
k <- kmeans(x, centers = 2, nstart=20)
k
```

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

Cluster means:
          x         y
1 -3.002555  2.734328
2  2.734328 -3.002555

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 1 1

Within cluster sum of squares by cluster:
[1] 69.22348 69.22348
 (between_SS / total_SS =  87.7 %)

Available components:
```

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

```
#the result tells a lot of data.
```

What's in this result object??

```
attributes(k)
```

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

$class
[1] "kmeans"
```

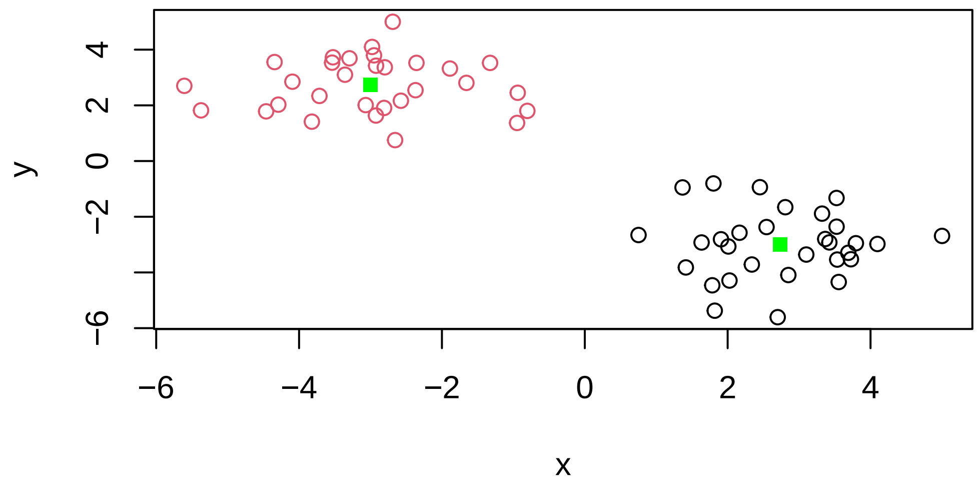
To get in-depth data about the clusters:

```
k$centers
```

```
          x         y
1 -3.002555  2.734328
2  2.734328 -3.002555
```

> Q. Plot your data `x` showing your clustering result and the center point of each cluster.

```
k <- kmeans(x, centers = 2, nstart=20)

plot(x, col=(k$cluster))
points(k$centers, pch=15, col="green")
```



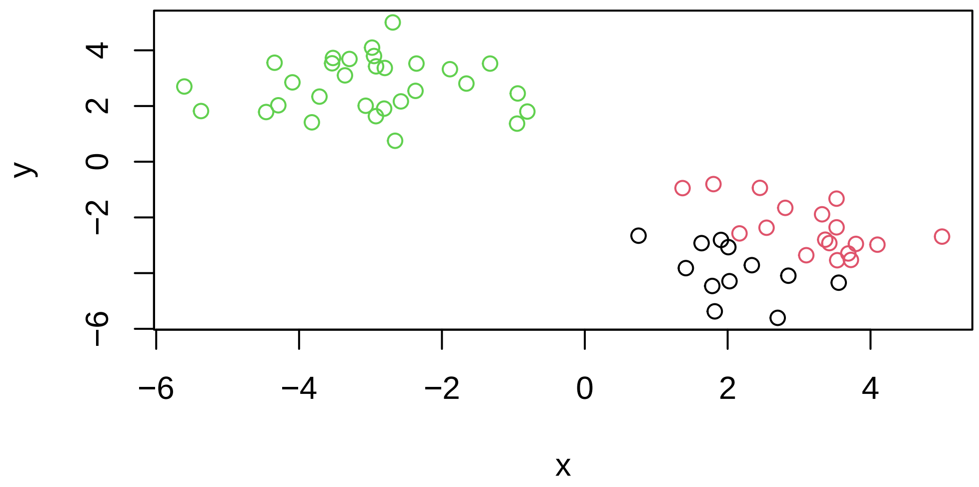
Q. Run kmeans and cluster into 3 groups and plot the results.

```r
k2 <- kmeans(x, centers = 3)
k2
```

```
K-means clustering with 3 clusters of sizes 12, 18, 30

Cluster means:
          x         y
1  2.065841 -3.929930
2  3.179985 -2.384305
3 -3.002555  2.734328

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

Within cluster sum of squares by cluster:
[1] 16.00785 27.07769 69.22348
 (between_SS / total_SS =  90.0 %)

Available components:
```

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

```
plot(x, col=k2$cluster)
```



Lets look at something:

Larger number of clusters

```
k$tot.withinss
```

```
[1] 138.447
```

```
k2$tot.withinss
```

```
[1] 112.309
```

The big limitation to kmeans() is that it **imposes structure on your data** that you ask for in the first place.

Hierarchical clustering

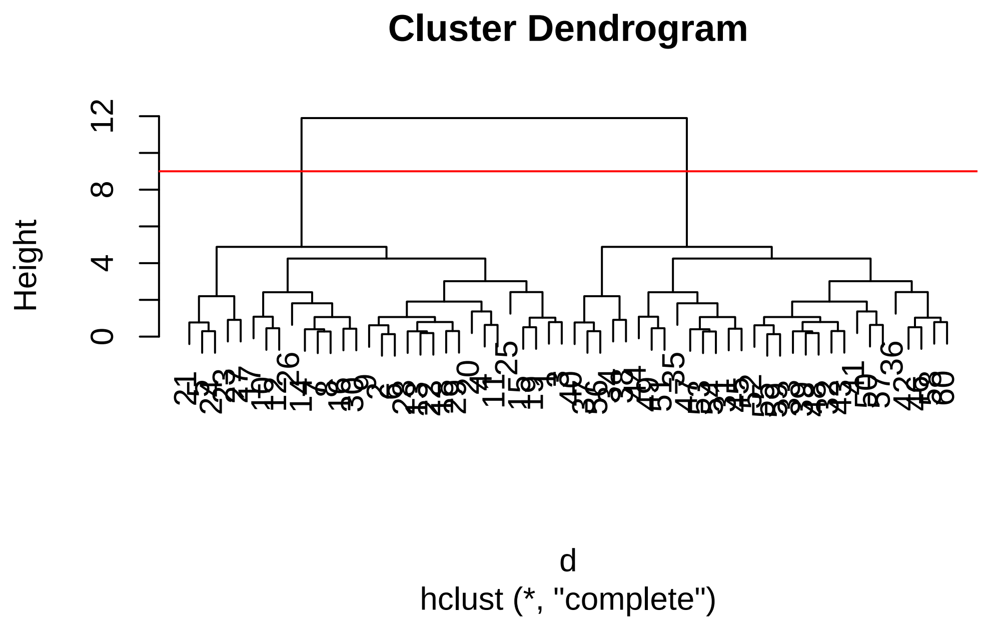
`hclust()` wont work with just the data, if you put it. The data needs to come from `dist()` (a distance matrix).

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

hc
```

```

Call:
hclust(d = d)

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

hc by itself doesn't give you much information so we use a new plot. Like this:

```r
plot(hc)
abline(h=9, col="red")
```

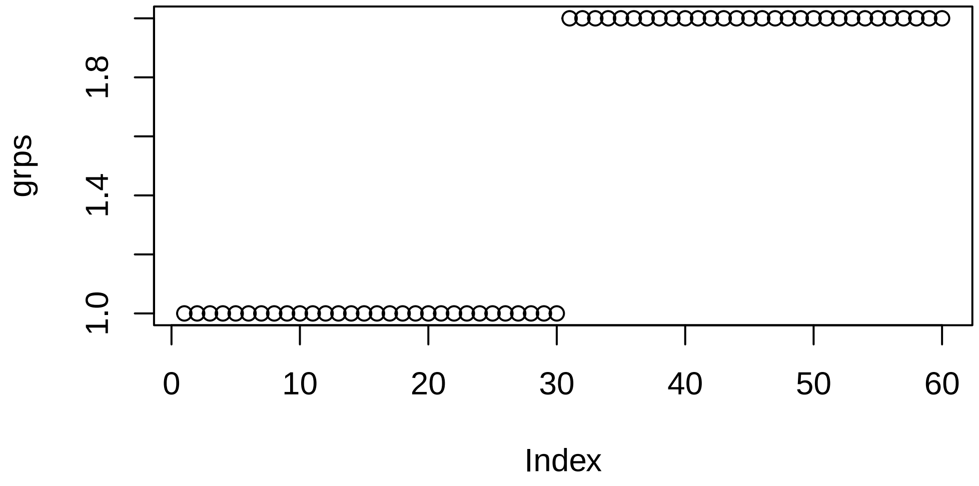


Let's now get the cluster membership vector we need to cut the tree at a given height we choose to seperate out some clusters. The function to do this calls for `cutree()`.

```
grps <- cutree(hc, h=9)
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
```

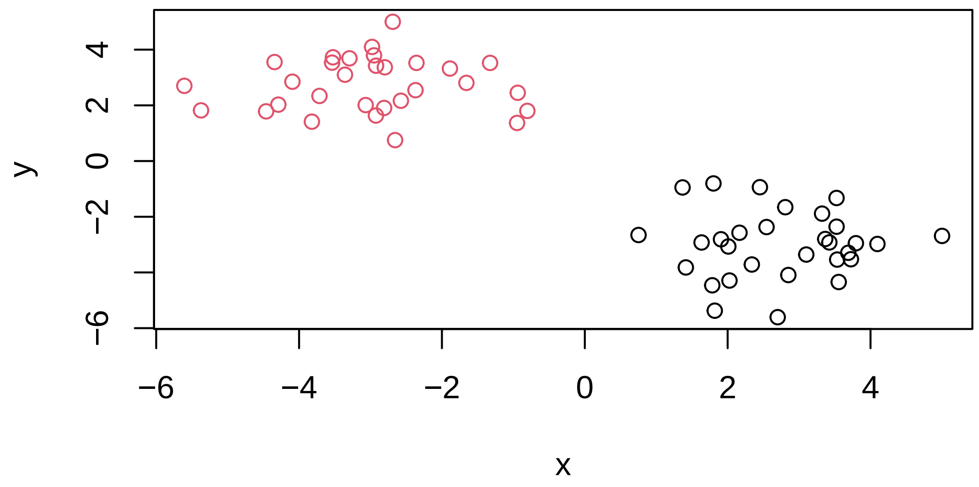
```
plot(grps)
```



```
# You can use cluster (specify the no. of clusters) or by height (h).
```

Q. Plot the results of data (x) and color our hclust.

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



Principal Component Analysis

We will start with PCA of a tiny dataset from UK gov.

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

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
Fresh_potatoes          720   874      566      1033
Fresh_Veg               253   265      171       143
Other_Veg               488   570      418       355
Processed_potatoes      198   203      220       187
Processed_Veg           360   365      337       334
```

```
Fresh_fruit               1102  1137       957        674
Cereals                   1472  1582      1462       1494
Beverages                   57    73        53         47
Soft_drinks               1374  1256      1572       1506
Alcoholic_drinks           375   475       458        135
Confectionery               54    64        62         41
```

One useful plot from the lab :

> Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

Answer:17 rows and 4 columns. `dim(x)` can be used as shown below.

**NOTE: for this answer, I already removed the first column X that showed the food macros.

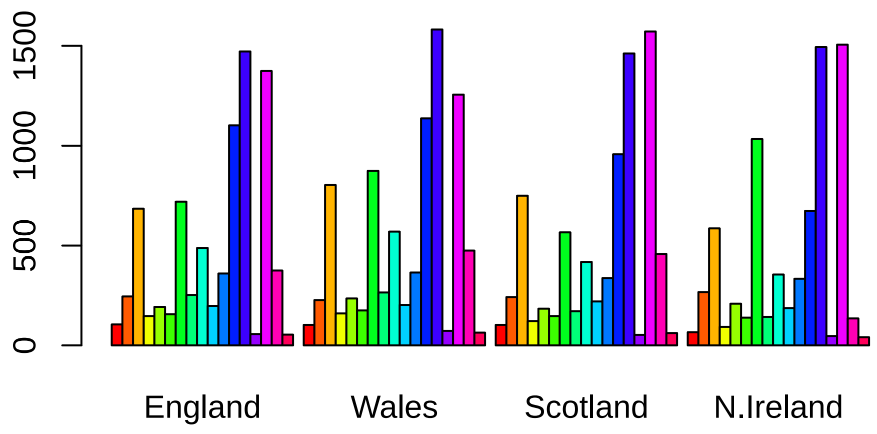
```
dim(x)
```

```
[1] 17  4
```

> Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

Answer: I like the option where I put the argument when I am reading the file such as: `x <- read.csv(url, row.names = 1)`. I think this is an easy way compared to the other one where I have to assign multiple variables.

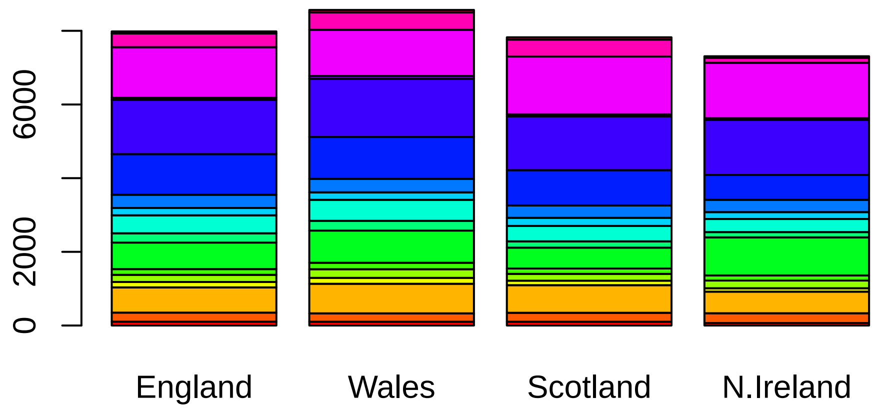
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



Q3.Changing what optional argument in the above barplot() function results in the following plot?

Answer: change the arguement for `beside()` to F as shown below:

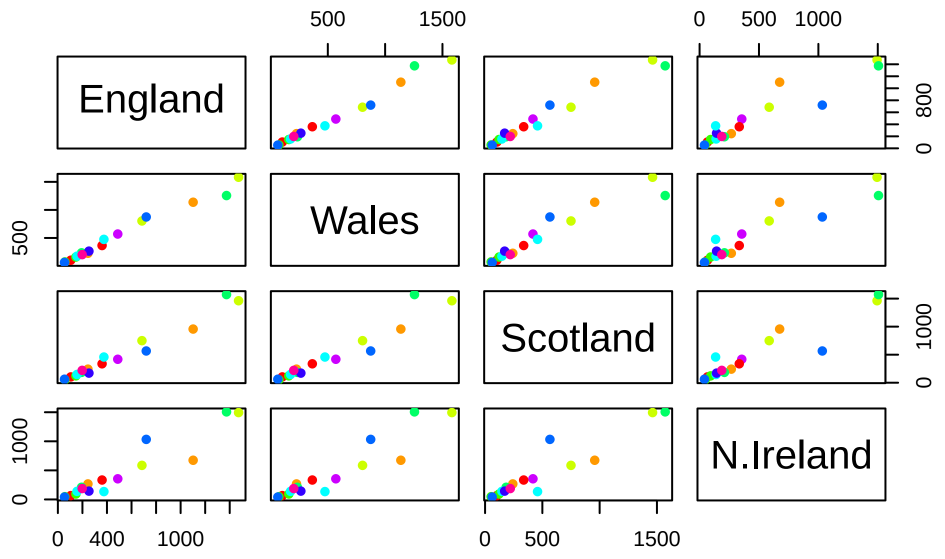
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



> Q5: Generating all pairwise plots may help somewhat. Can you make sense of the following code and resulting figure? What does it mean if a given point lies on the diagonal for a given plot?

Answer: If points lie on the diagnols it means that the consumption of that food catergory is ideally same for the countries on x and y axis. In other words, the values of the variables are identical for countries.

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



```
# But this also doesnt give you much data
```

> Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set?

Answer: They consume the highest fresh potatoes compared to other countries.

Enter PCA

The main fucntion to do PCA in "base" R is called `prcomp()`

It wants our foods as the column and the countries as the rows.so we transpose it using `t()`.

```
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
```

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

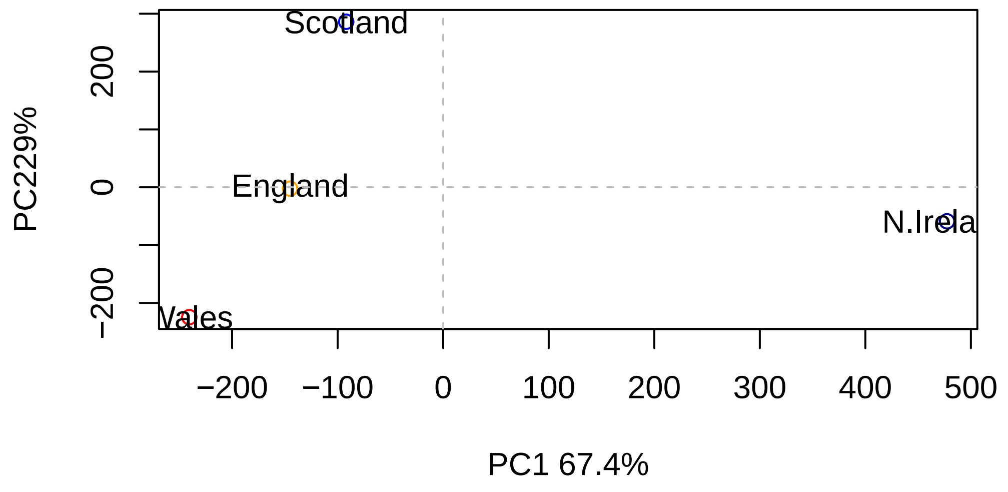
> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.
>
> Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

Answer as shown on the plots: **note: I have added color to the points on the graph.

```
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
```

```
plot(pca$x[,1], pca$x[,2], xlab = "PC1 67.4%", ylab = "PC229%", col=c("orange", "red", "bl
text(pca$x[,1], pca$x[,2], colnames(x))

abline(h=0, col="gray", lty=2)
abline(v=0, col="gray", lty=2)
```



Variable Loading

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

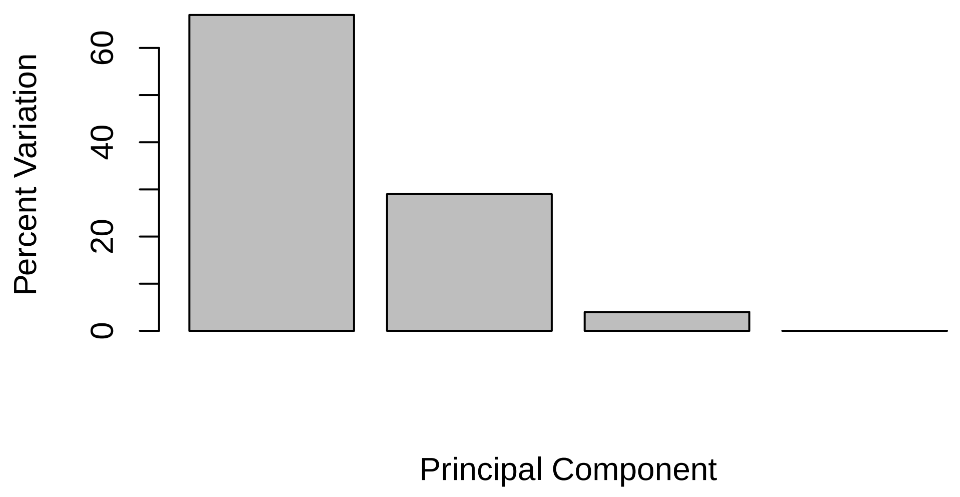
```
[1] 67 29  4  0
```

```
z <- summary(pca)
z$importance
```

```
                             PC1       PC2      PC3          PC4
Standard deviation     324.15019 212.74780 73.87622 2.921348e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

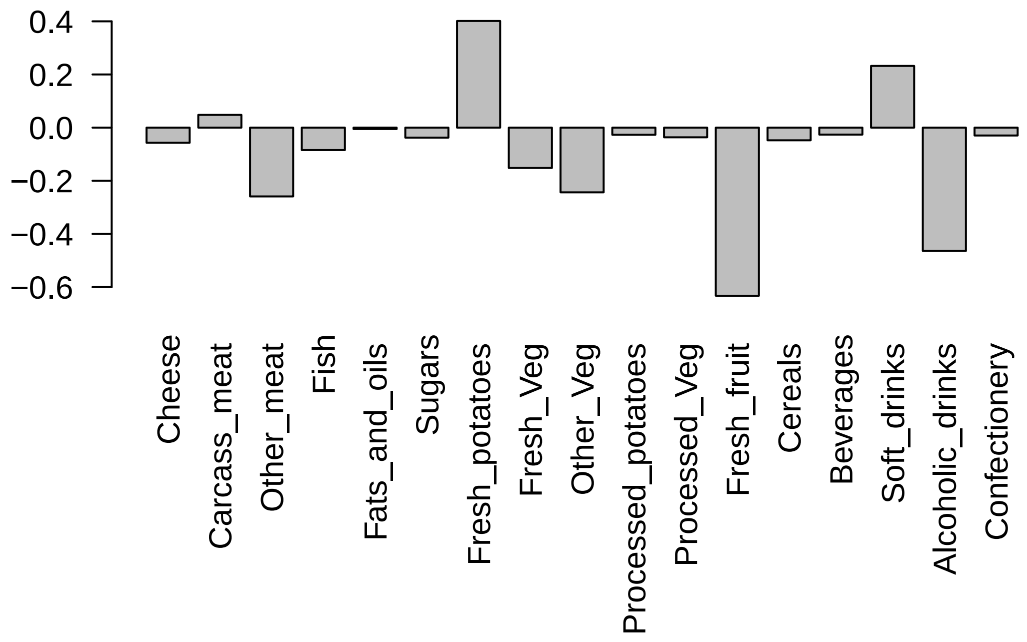
Plotting it:

```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



** Note: PC1 and PC2 account for the most variance in the data set. Therefore, they can be used to make the graphs.

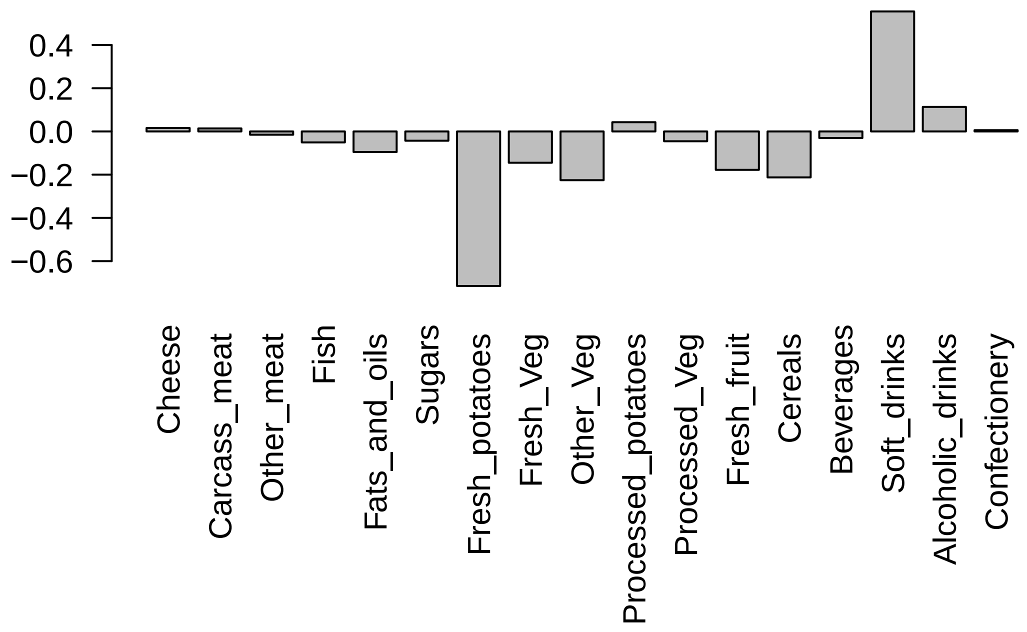
```r
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



> Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about?

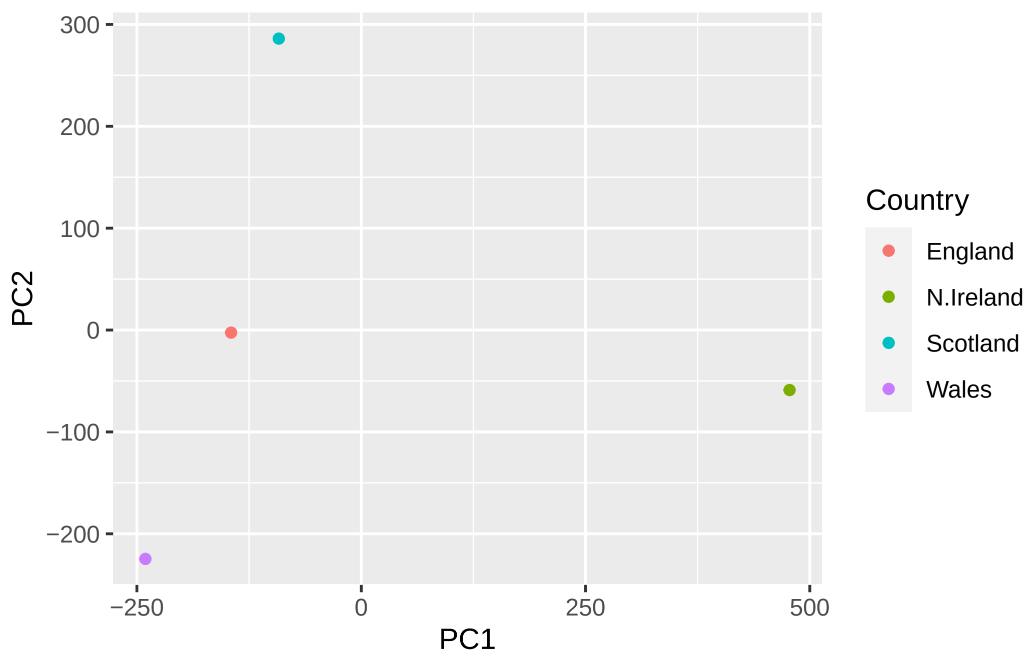
Answer: Soft drinks as shown by positive relation in the graph below.

```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```



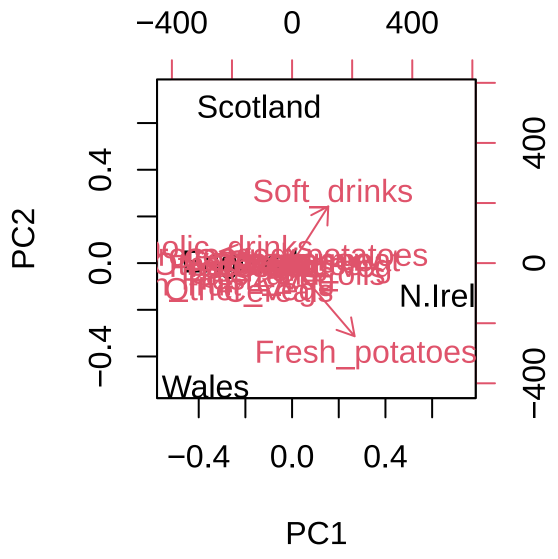
using ggplot

```
library(ggplot2)

df <- as.data.frame(pca$x)
df_lab <- tibble::rownames_to_column(df, "Country")

# Our first basic plot
ggplot(df_lab) +
  aes(PC1, PC2, col=Country) +
  geom_point()
```



300
200
100
0
−100
−200
PC2
−250
0
250
500
PC1
Country
England
N.Ireland
Scotland
Wales

```r
biplot(pca)
```



−400
0
400
Scotland
Soft_drinks
Fresh_potatoes
N.Irel
Wales
PC2
PC1
0.4
0.0
−0.4

Q10: How many genes and samples are in this data set?

I think this question was covered on the other day. Answer: 100 genes

```r
url2 <- "https://tinyurl.com/expression-CSV"
rna.data <- read.csv(url2, row.names=1)
head(rna.data)
```

```
       wt1 wt2  wt3  wt4 wt5 ko1 ko2 ko3 ko4 ko5
gene1  439 458  408  429 420  90  88  86  90  93
gene2  219 200  204  210 187 427 423 434 433 426
gene3 1006 989 1030 1017 973 252 237 238 226 210
gene4  783 792  829  856 760 849 856 835 885 894
gene5  181 249  204  244 225 277 305 272 270 279
gene6  460 502  491  491 493 612 594 577 618 638
```