

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:

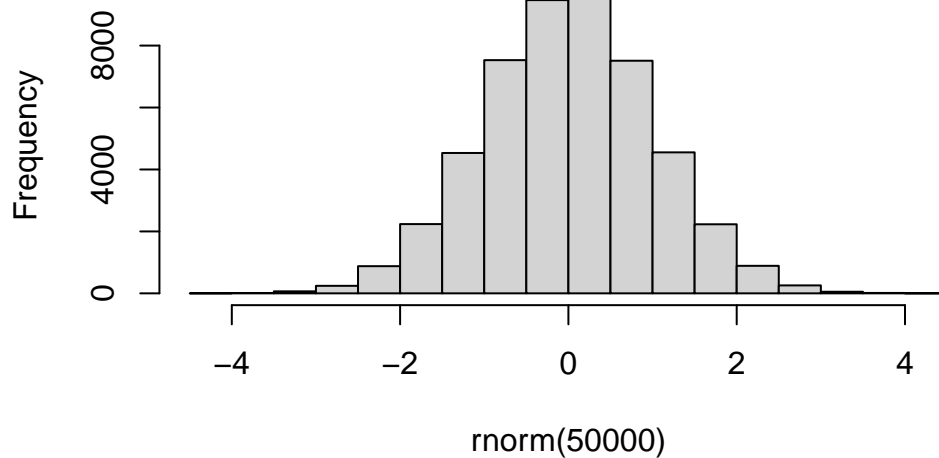
This gives random numbers or n; Its a way to make up random data:

```
runif(10)
```

```
[1] 0.6400587 -0.3521043 0.1034658 -1.1773587 1.3788333 -0.5989850  
[7] -0.4470089 1.3291616 -0.3579553 1.4229070
```

```
hist( runif(50000))
```

Histogram of rnorm(50000)



```
# mean will be 0 and sd is 1. This is the default.  
# 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] 2.6088917 3.6986128 4.1292694 3.6305963 4.2994736 2.7816797  
[7] 3.6622169 2.1059249 2.9430145 0.8395056 2.0895064 4.0853259  
[13] 0.5940555 2.1335508 4.8949444 3.9940823 2.9316474 3.6719410  
[19] 2.1050954 3.4731548 3.1773645 5.4769479 2.6746434 2.8690696  
[25] 2.2660432 1.9160394 4.5274297 4.5303955 4.2614649 1.7172860  
[31] -3.6560152 -3.5643001 -3.0195455 -2.7161073 -2.5076853 -1.0477904  
[37] -3.8751460 -3.2919489 -3.4595506 -3.8062196 -3.9509458 -2.9793642  
[43] -4.0961631 -3.2221439 -3.7828336 -3.8531873 -3.6619455 -2.1255266  
[49] -4.0210886 -1.0917759 -4.6526076 -2.9867599 -2.0363258 -4.1800330  
[55] -2.0228602 -1.7486847 -1.7649620 -5.3481551 -3.3756427 0.8080443
```

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

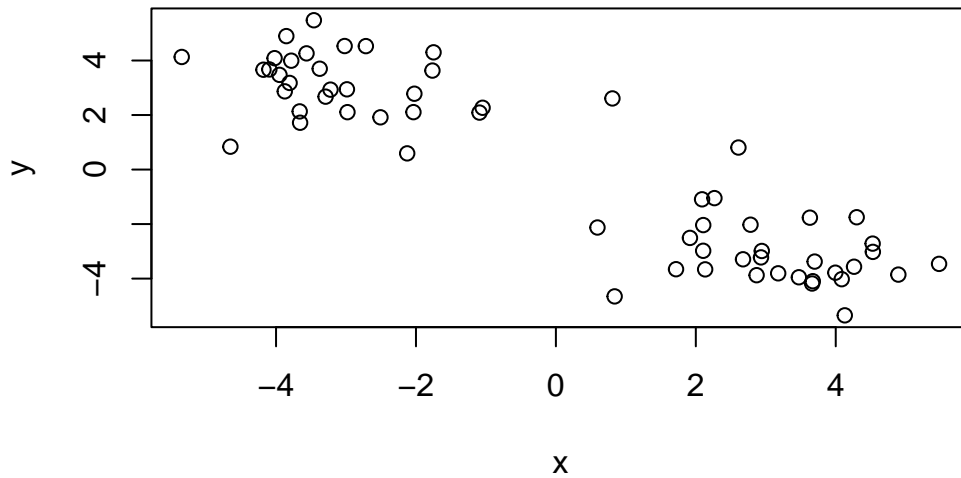
```
rev.tmp <- rev(tmp)
```

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

	x	y
[1,]	2.6088917	0.8080443
[2,]	3.6986128	-3.3756427
[3,]	4.1292694	-5.3481551
[4,]	3.6305963	-1.7649620
[5,]	4.2994736	-1.7486847
[6,]	2.7816797	-2.0228602
[7,]	3.6622169	-4.1800330
[8,]	2.1059249	-2.0363258
[9,]	2.9430145	-2.9867599
[10,]	0.8395056	-4.6526076
[11,]	2.0895064	-1.0917759
[12,]	4.0853259	-4.0210886
[13,]	0.5940555	-2.1255266
[14,]	2.1335508	-3.6619455
[15,]	4.8949444	-3.8531873
[16,]	3.9940823	-3.7828336
[17,]	2.9316474	-3.2221439
[18,]	3.6719410	-4.0961631
[19,]	2.1050954	-2.9793642
[20,]	3.4731548	-3.9509458
[21,]	3.1773645	-3.8062196
[22,]	5.4769479	-3.4595506
[23,]	2.6746434	-3.2919489
[24,]	2.8690696	-3.8751460
[25,]	2.2660432	-1.0477904
[26,]	1.9160394	-2.5076853
[27,]	4.5274297	-2.7161073
[28,]	4.5303955	-3.0195455
[29,]	4.2614649	-3.5643001
[30,]	1.7172860	-3.6560152
[31,]	-3.6560152	1.7172860
[32,]	-3.5643001	4.2614649
[33,]	-3.0195455	4.5303955
[34,]	-2.7161073	4.5274297
[35,]	-2.5076853	1.9160394

```
[36,] -1.0477904  2.2660432
[37,] -3.8751460  2.8690696
[38,] -3.2919489  2.6746434
[39,] -3.4595506  5.4769479
[40,] -3.8062196  3.1773645
[41,] -3.9509458  3.4731548
[42,] -2.9793642  2.1050954
[43,] -4.0961631  3.6719410
[44,] -3.2221439  2.9316474
[45,] -3.7828336  3.9940823
[46,] -3.8531873  4.8949444
[47,] -3.6619455  2.1335508
[48,] -2.1255266  0.5940555
[49,] -4.0210886  4.0853259
[50,] -1.0917759  2.0895064
[51,] -4.6526076  0.8395056
[52,] -2.9867599  2.9430145
[53,] -2.0363258  2.1059249
[54,] -4.1800330  3.6622169
[55,] -2.0228602  2.7816797
[56,] -1.7486847  4.2994736
[57,] -1.7649620  3.6305963
[58,] -5.3481551  4.1292694
[59,] -3.3756427  3.6986128
[60,]  0.8080443  2.6088917
```

```
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	3.136306	-3.034576
2	-3.034576	3.136306

Clustering vector:

[illegible]

Within cluster sum of squares by cluster:

```
[1] 85.12047 85.12047
(between_SS / total_SS = 87.0 %)
```

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:

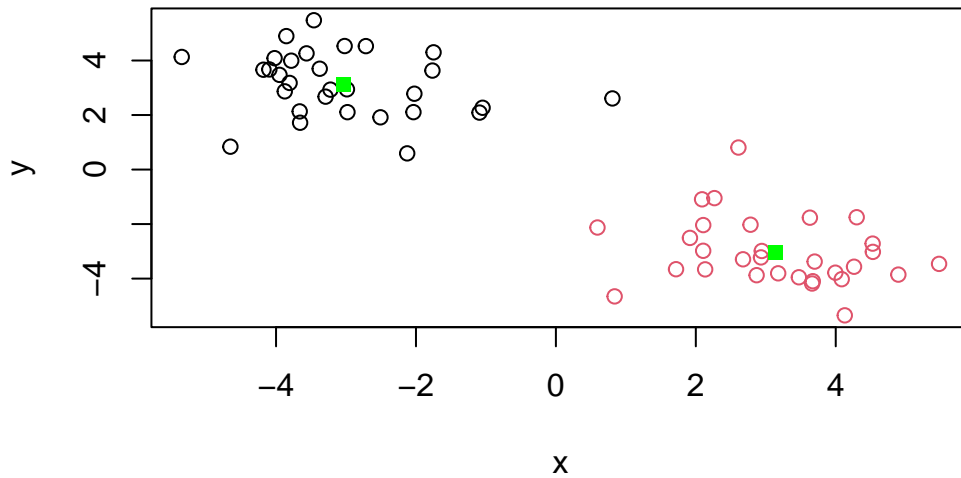
```
k$centers
```

```
      x      y
1  3.136306 -3.034576
2 -3.034576  3.136306
```

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.

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

K-means clustering with 3 clusters of sizes 30, 10, 20

Cluster means:

	x	y
1	3.136306	-3.034576
2	-1.651693	2.439731
3	-3.726017	3.484593

Clustering vector:

[1] 1 3 3 3 3 2 2 3 3
[39] 3 3 3 2 3 3 3 3 2 3 2 3 3 2 3 2 2 2 3 3 2

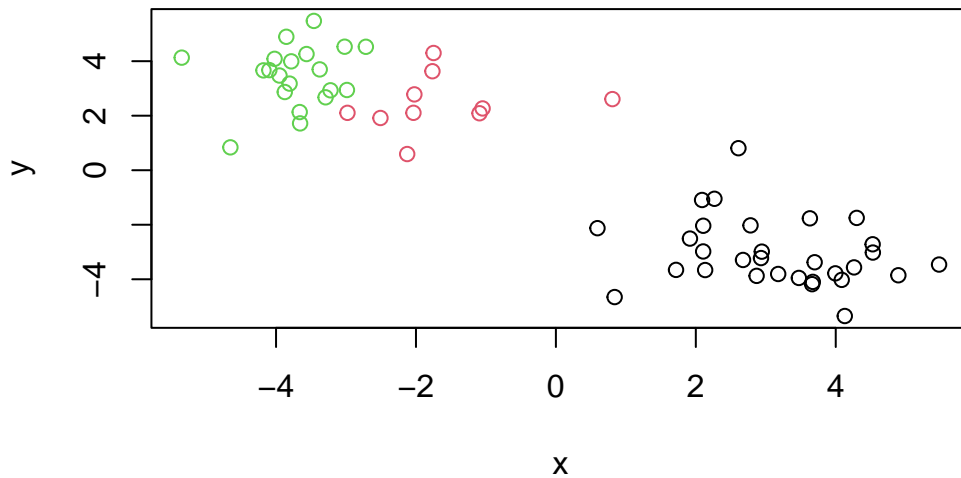
Within cluster sum of squares by cluster:

```
[1] 85.12047 18.83576 30.32099
      (between_SS / total_SS =  89.8 %)
```

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] 170.2409
```

```
k2$tot.withinss
```

```
[1] 134.2772
```

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

Hierarchical clustering

`hclust()` won't work with just the data, if you put it. The data needs to come from `dist()` (a distance matrix).

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

hc
```

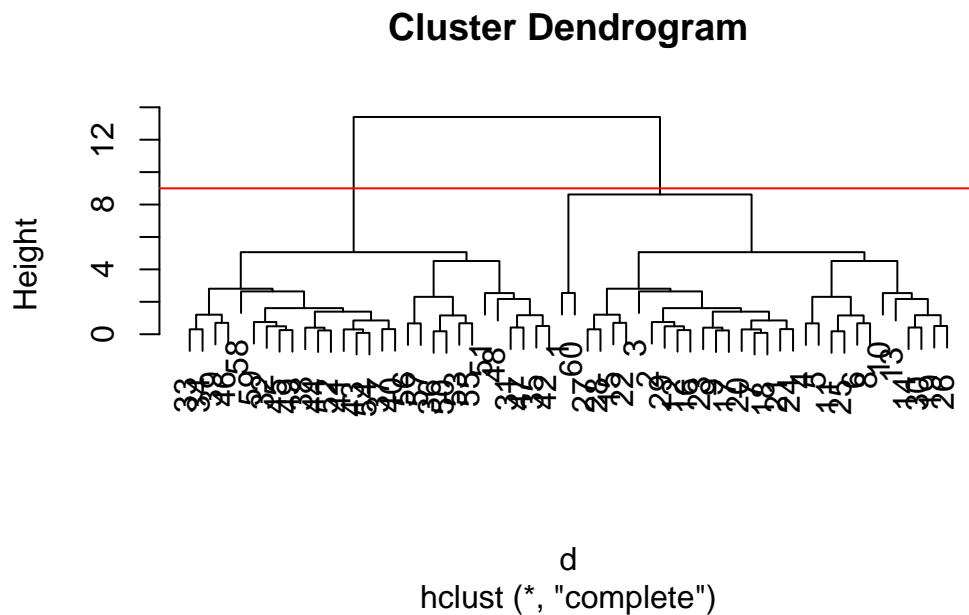
Call:

```
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:

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

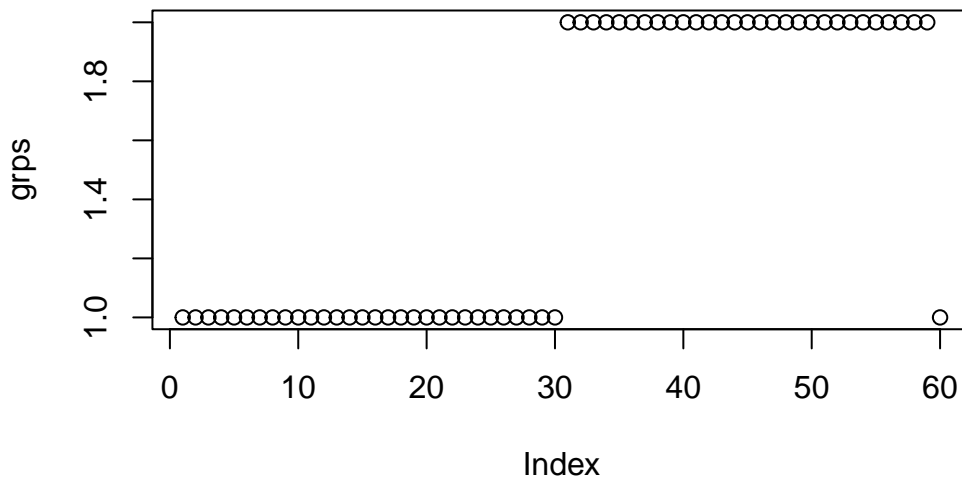


Let's now get the cluster membership vector we need to cut the tree at a given height we choose to separate out some clusters. The function to do this calls for `cutree()`.

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

[illegible]

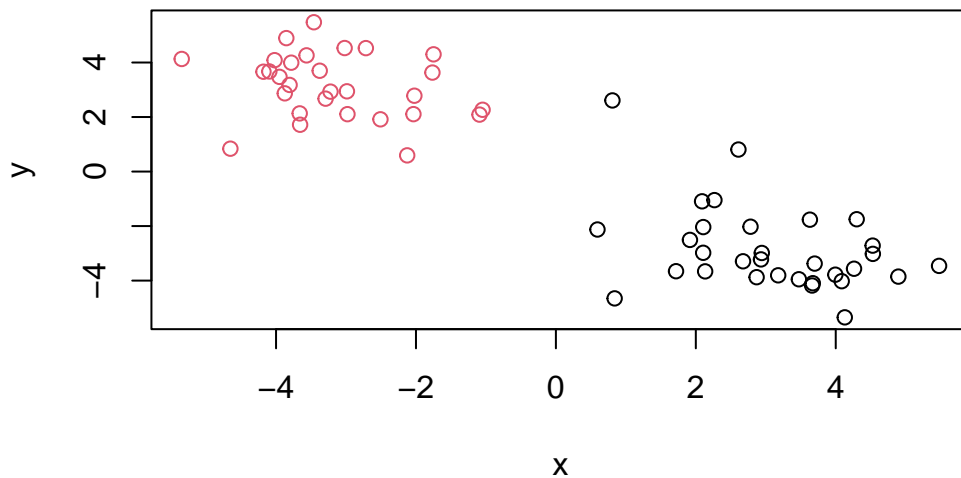
```
plot(grps)
```



```
# 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.

```
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.

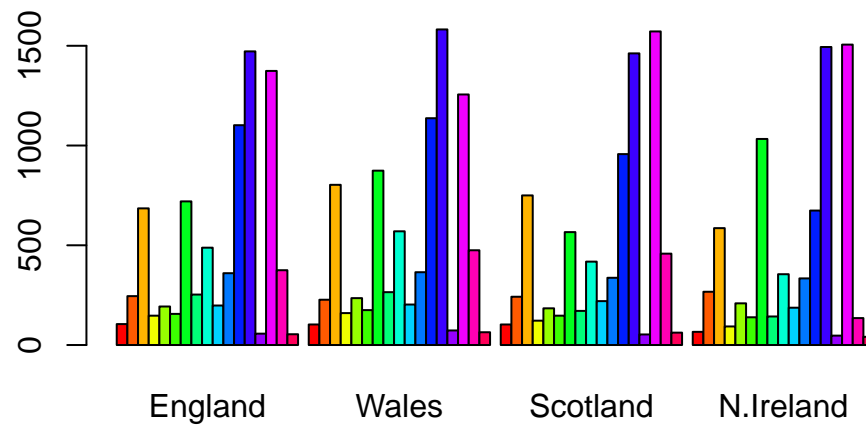
```
dim(x)
```

```
[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.

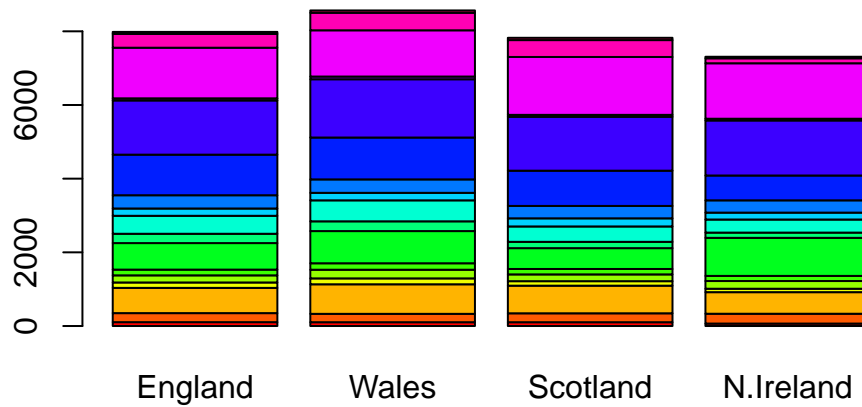
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

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

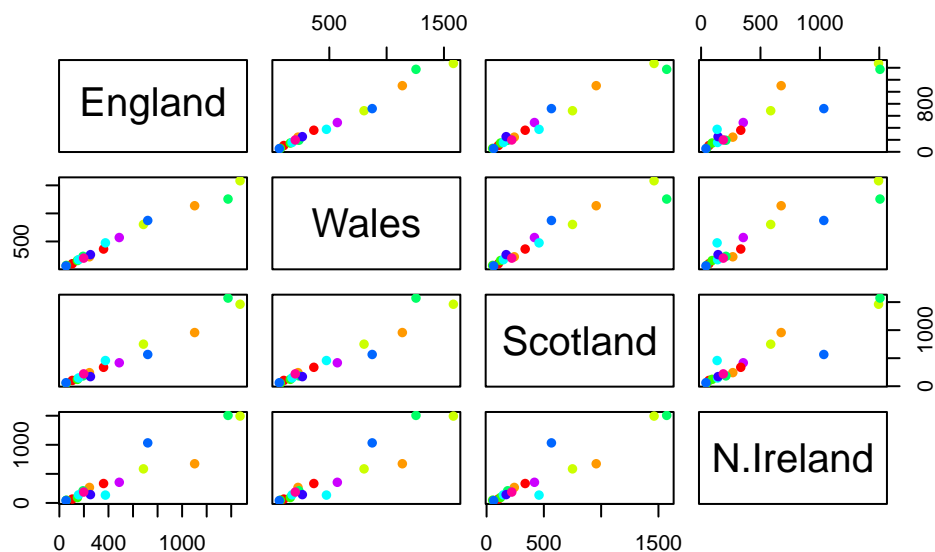
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



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 diagonals it means that the consumption of that food category 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 fuction 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.

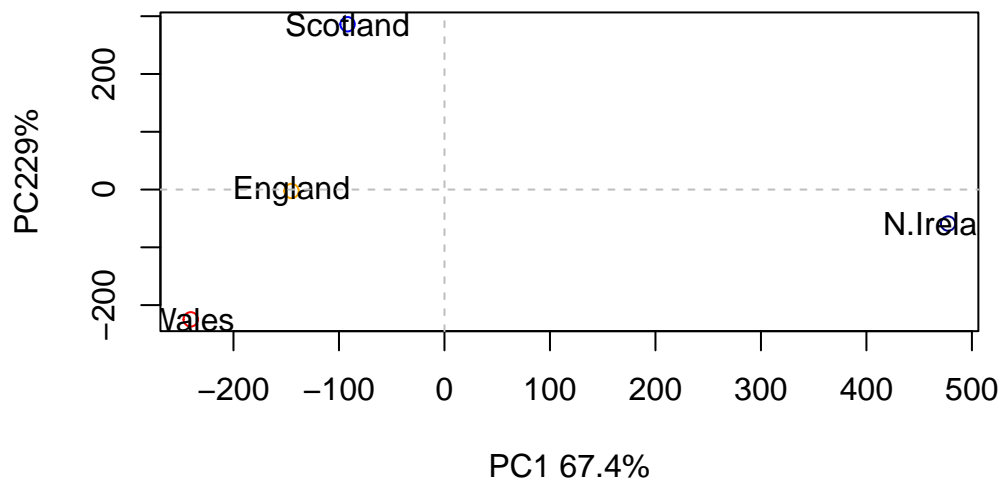
```
pca$x
```

	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", "blue"),  
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
```

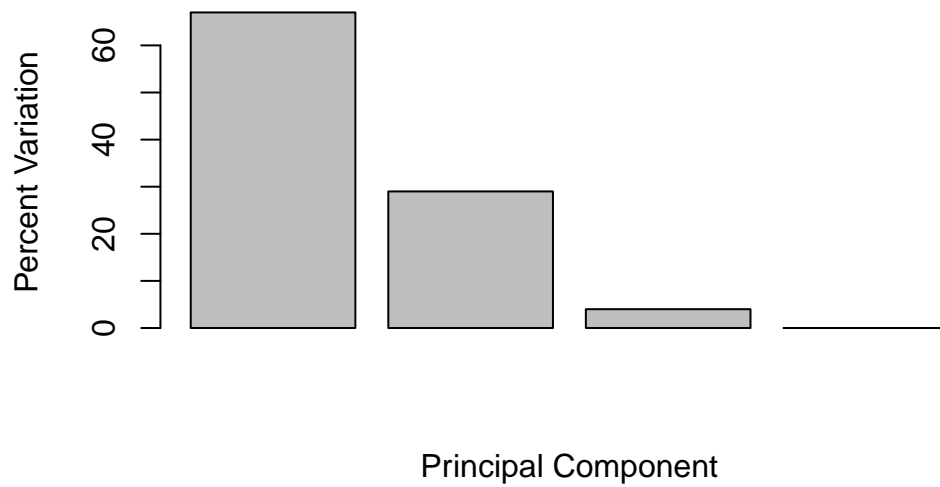
```
[1] 67 29  4  0
```

```
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

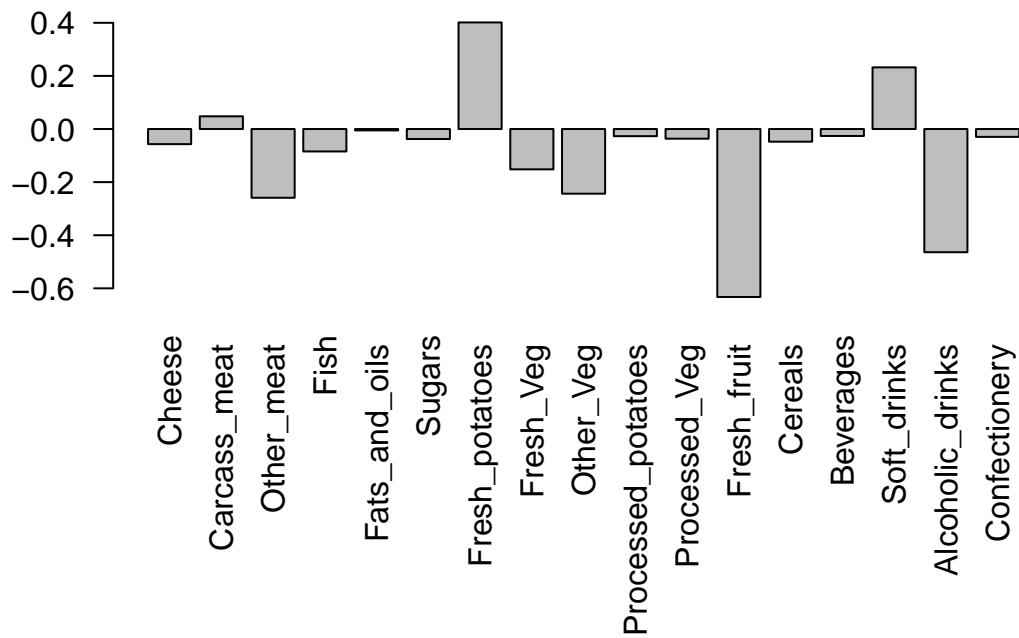
Plotting it:

```
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.

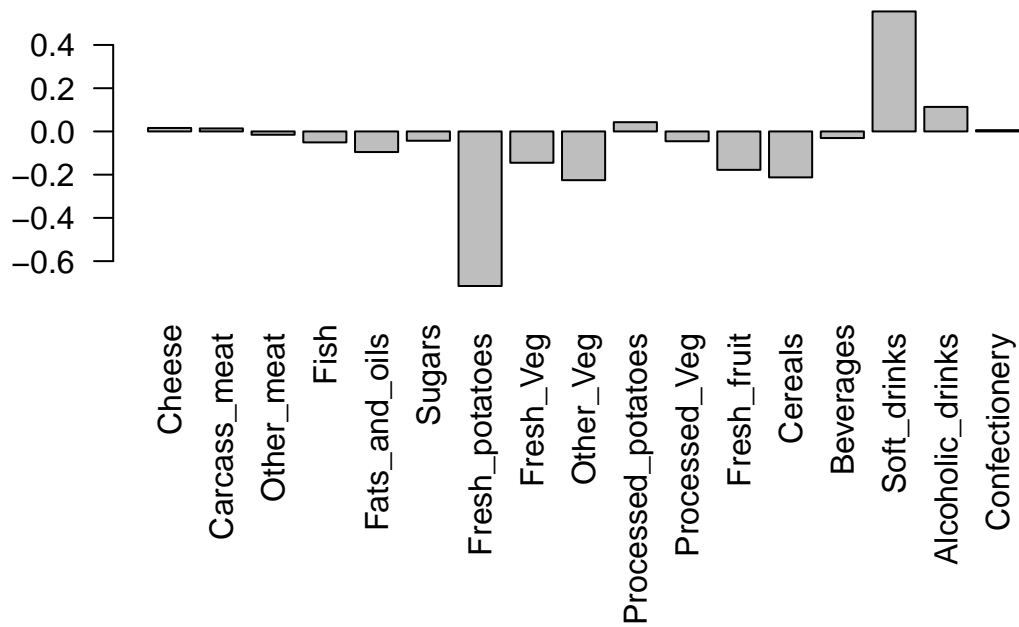
```
par(mar=c(10, 3, 0.35, 0))  
barplot( pca$rotation[,1], las=2 )
```



Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominently and what does PC2 mainly 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()
```


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

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

```
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