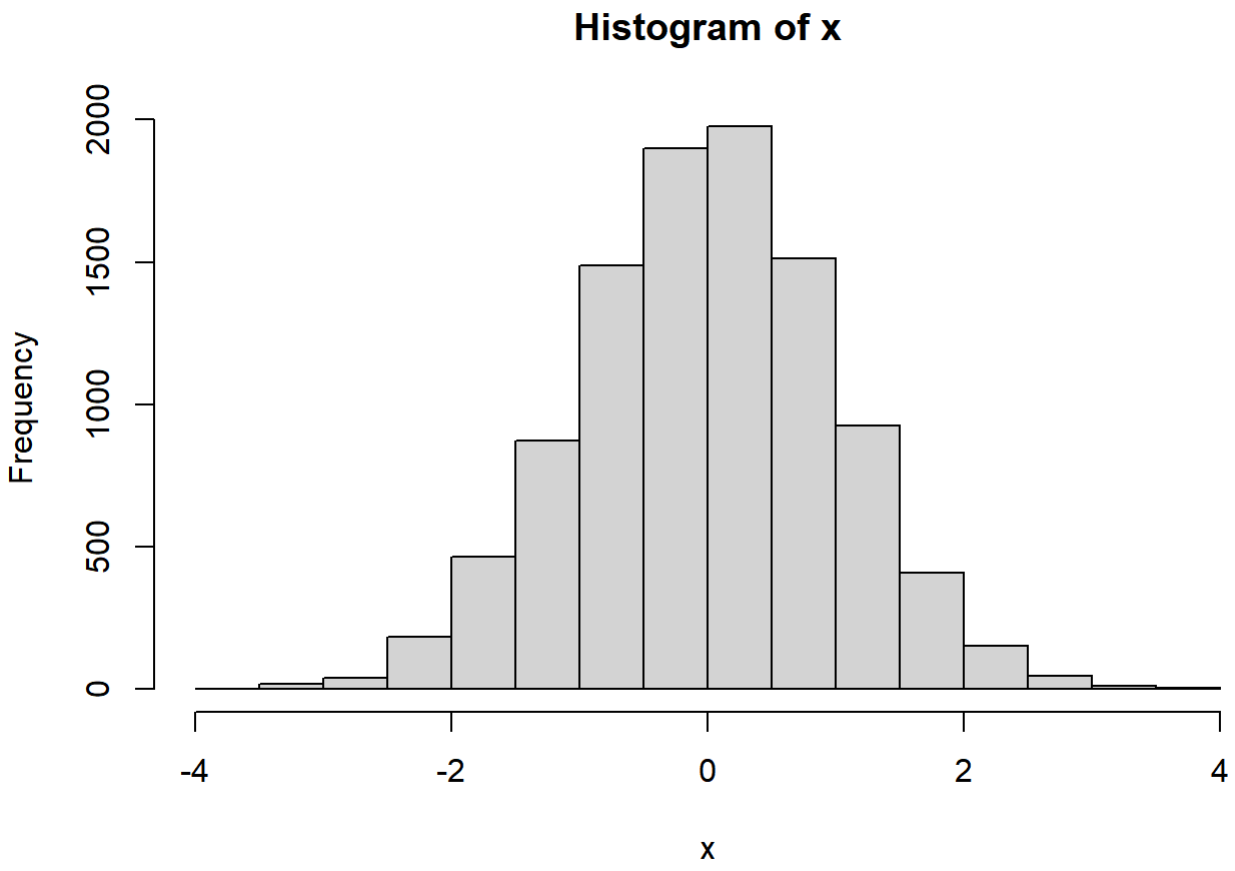
Class7ML

AUTHOR
Nancy Lin

K-means clustering

First we will test how this works in R with some made up data.

```
#RNORM() Random generation of number

x <- rnorm(10000)
hist(x)
```



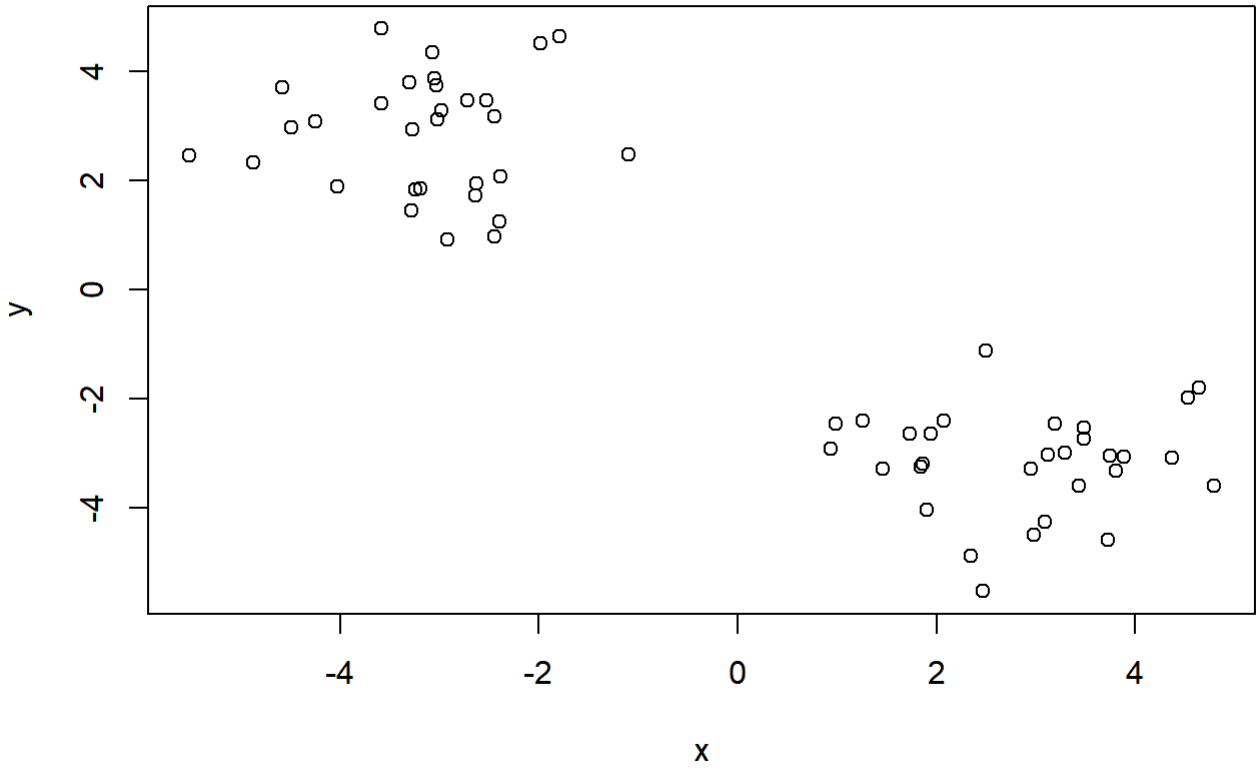
Let's make some numbers center on -3

```
rev(c("a","b"))
```

```
[1] "b" "a"
```

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

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



NOw let's see how 'kmeans' works with this data...

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

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

Cluster means:
          x         y
1 -3.145454  2.859793
2  2.859793 -3.145454

Clustering vector:
 [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 2 2 2

Within cluster sum of squares by cluster:
[1] 60.54031 60.54031
 (between_SS / total_SS =  89.9 %)

Available components:
```

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

> Q. How many points are in each cluster?

```
km$size
```

```
[1] 30 30
```

> Q. What component of your result object details -cluster assignment/ membership -cluster center

```
km$centers
```

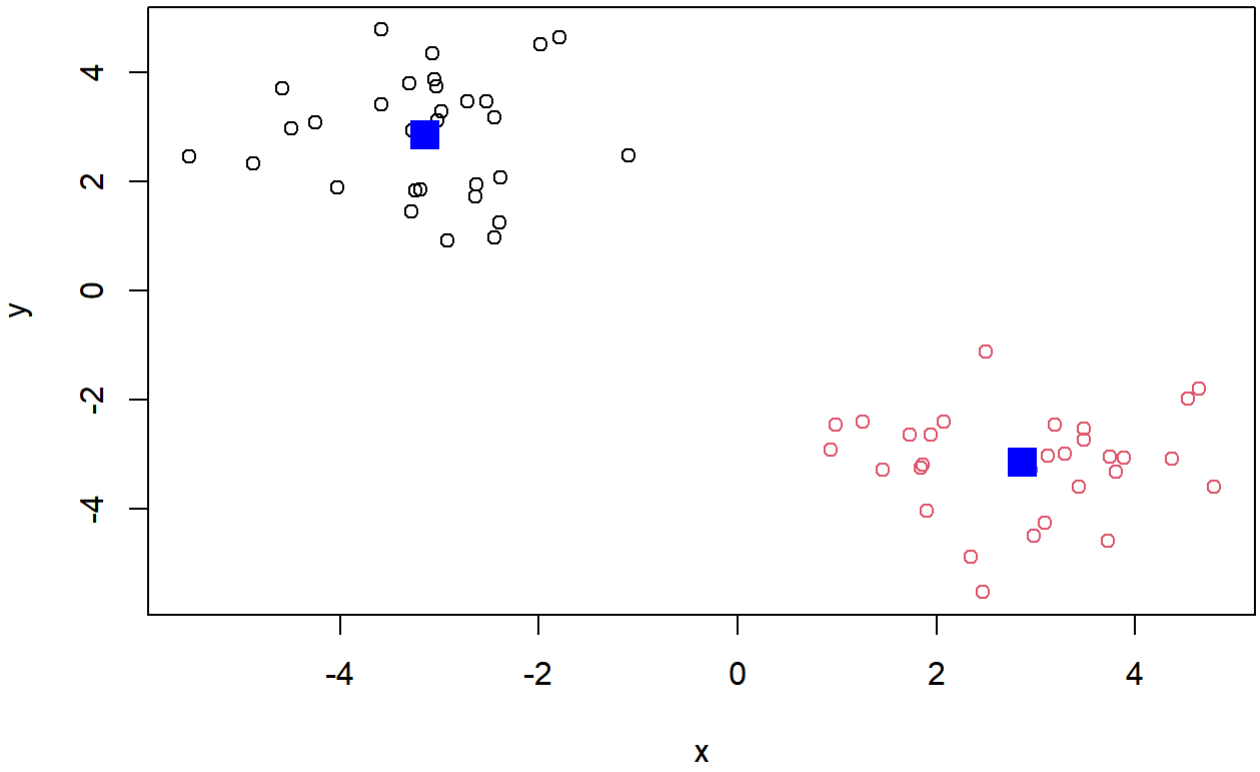
```
          x         y
1 -3.145454  2.859793
2  2.859793 -3.145454
```

```
km$cluster
```

```
 [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
```

> Q. Plot x colored by the kmeans cluster assignment and add cluster centers as blue points

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



Hierarchical Clustering

The 'hclust()' function in R performs hierarchical clustering.

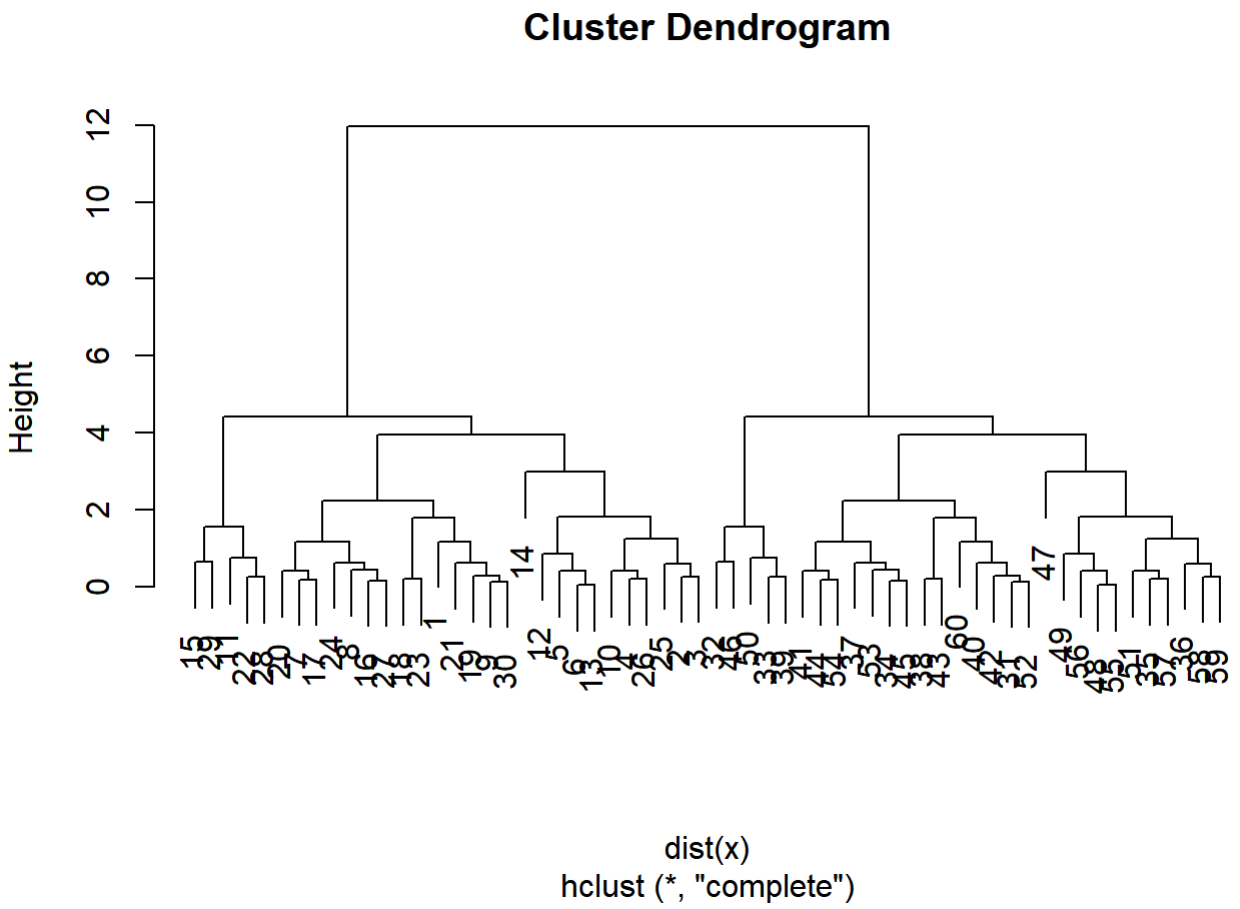
The 'hclust()' function requires an input distance matrix, which i can get from the 'dist()' function.

```
#cluster(dist(x))

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

```
Call:
hclust(d = dist(x))

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

There is a plot()method for hclust objects...

```
plot(hc)
```



Now to get my cluster membership vector I need to cut the tree to yield separate “branches” with the leaves on each branch being our clusters. To do this we us tge ‘cutree()’ function.

```
cutree(hc,h=8)
```

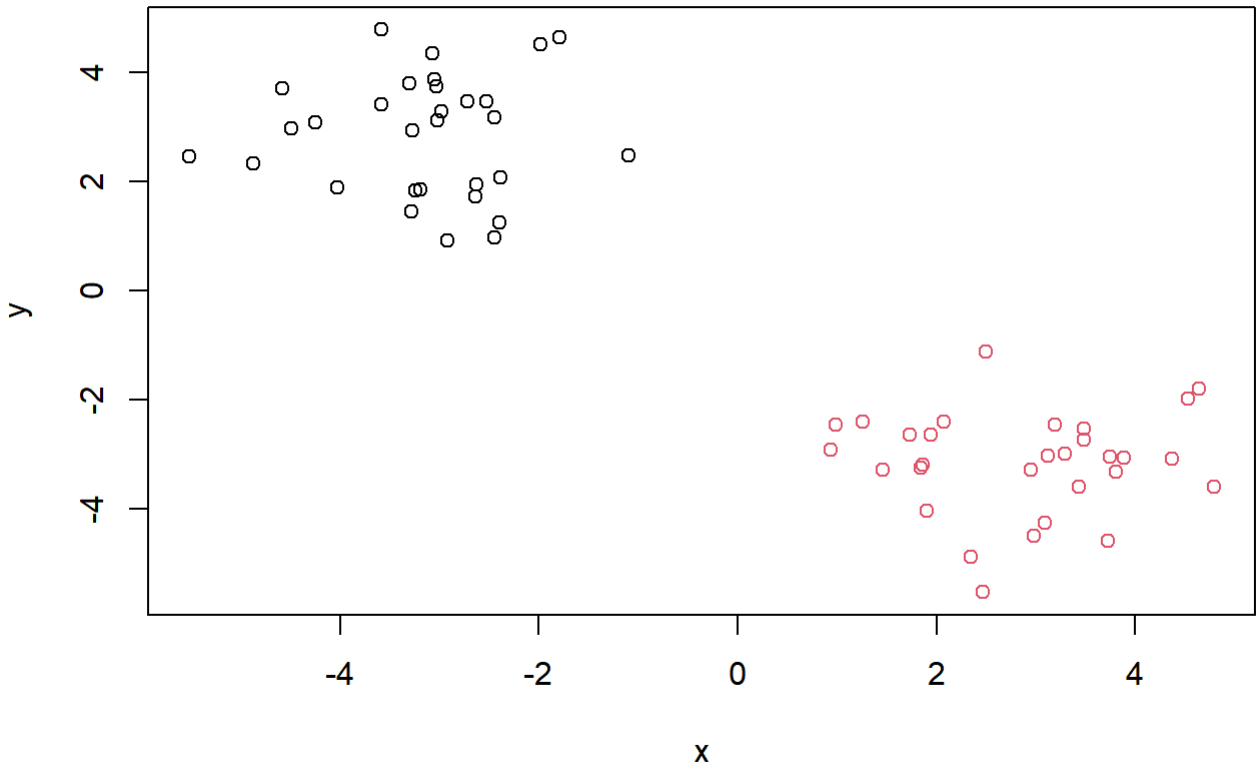
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Use ‘cutree()’ with a k=2 — Find the height for you

```
grps <- cutree(hc,k=2)
```

A plot of our data colored by our hclut grps:

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



Principal Component Analysis (PCA)

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

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

```
data.frame(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

There are 17 rows and 5 columns.

> Q2. FUnction view

```
## Preview the first 6 rows
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

```
# Note how the minus indexing works
rownames(x) <- x[,1]
x <- x[,-1]
head(x)
```

	Wales	Scotland	N.Ireland
105	103	103	66
245	227	242	267
685	803	750	586
147	160	122	93
193	235	184	209
156	175	147	139

Check dim:

```
dim(x)
```

```
[1] 17  3
```

alt approach:

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

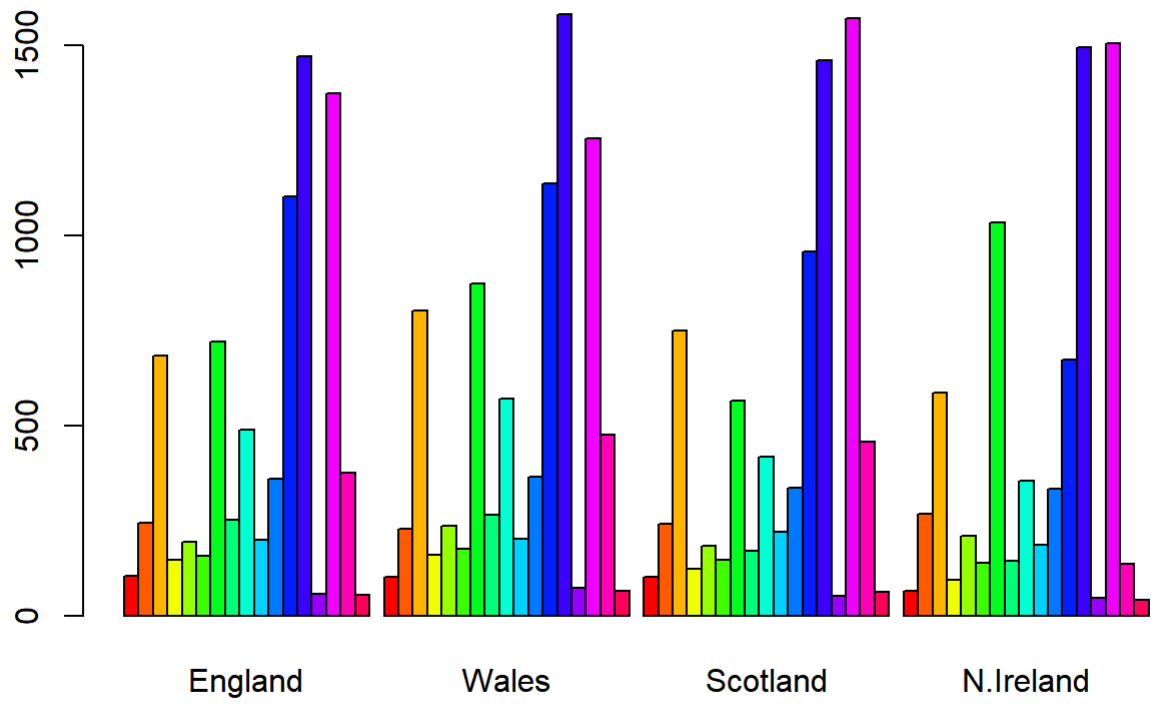
```
              England Wales Scotland N.Ireland
Cheese            105   103      103        66
Carcass_meat      245   227      242       267
Other_meat        685   803      750       586
Fish              147   160      122        93
Fats_and_oils     193   235      184       209
Sugars        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

> 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?

The second approach is better since it will not delete columns every time we run the function.

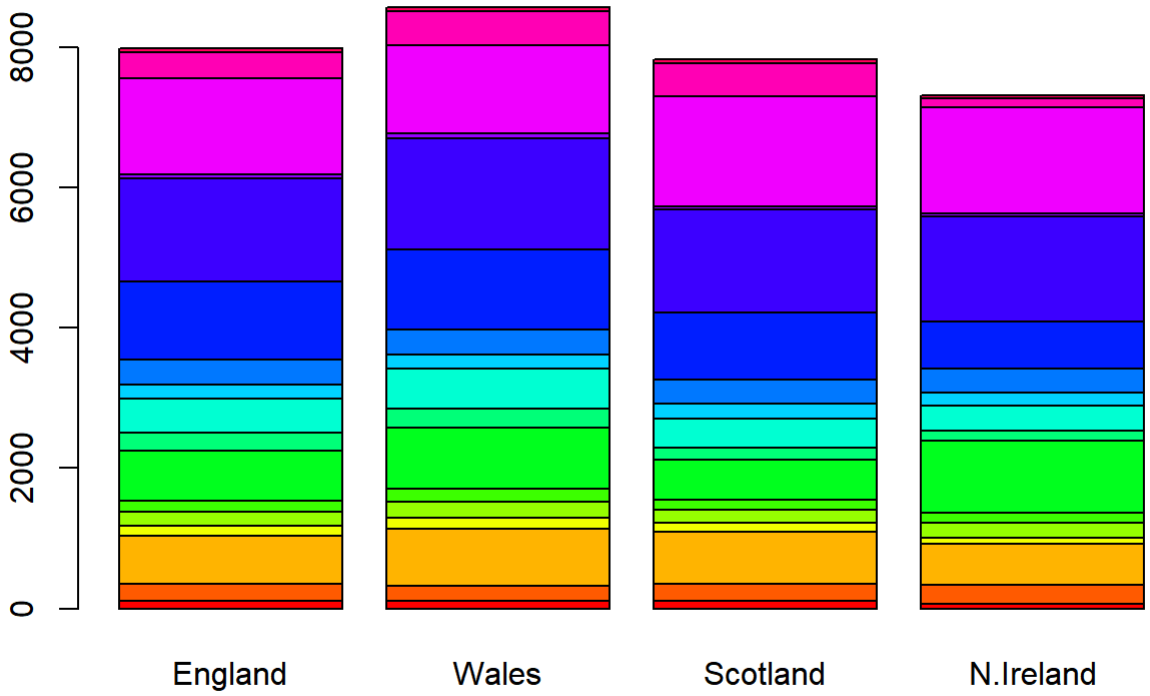
Let's make some not useful plot:

```
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?

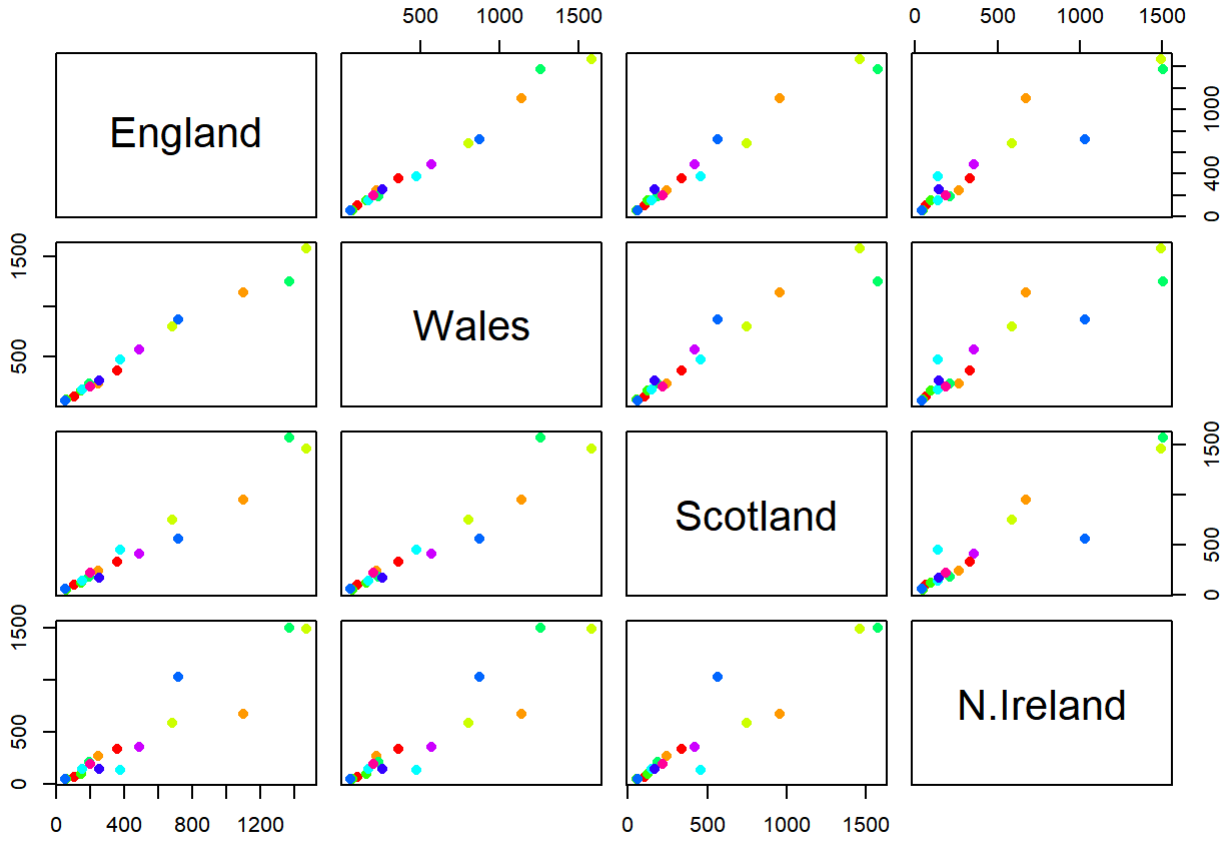
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



> 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?

We compare two countires at the same time. If a given point lies on the diagonal, it means the two countries eat the same amount for the type of food.

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



log 2 x>y=1 x<y=-1 below: more of it on x

While this is kind of useful, it takes work to dig into the details here to find out what is different in the countries.

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

The blue point.

PCA to the rescue

Principal Component Analysis can help where we have a lot of dimension being measure in the data set.

The main PCA function in base R is called 'prcomp()'.

The 'prcomp' function wants us input the transpose of out foodmatrix frame.

```
# t (x) filp x and Y
```

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

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 4.189e-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 above result shows PCA captures 67 % of the total variance in the original data in one PC and 96.5% in two PCs.

```
attributes(pca)
```

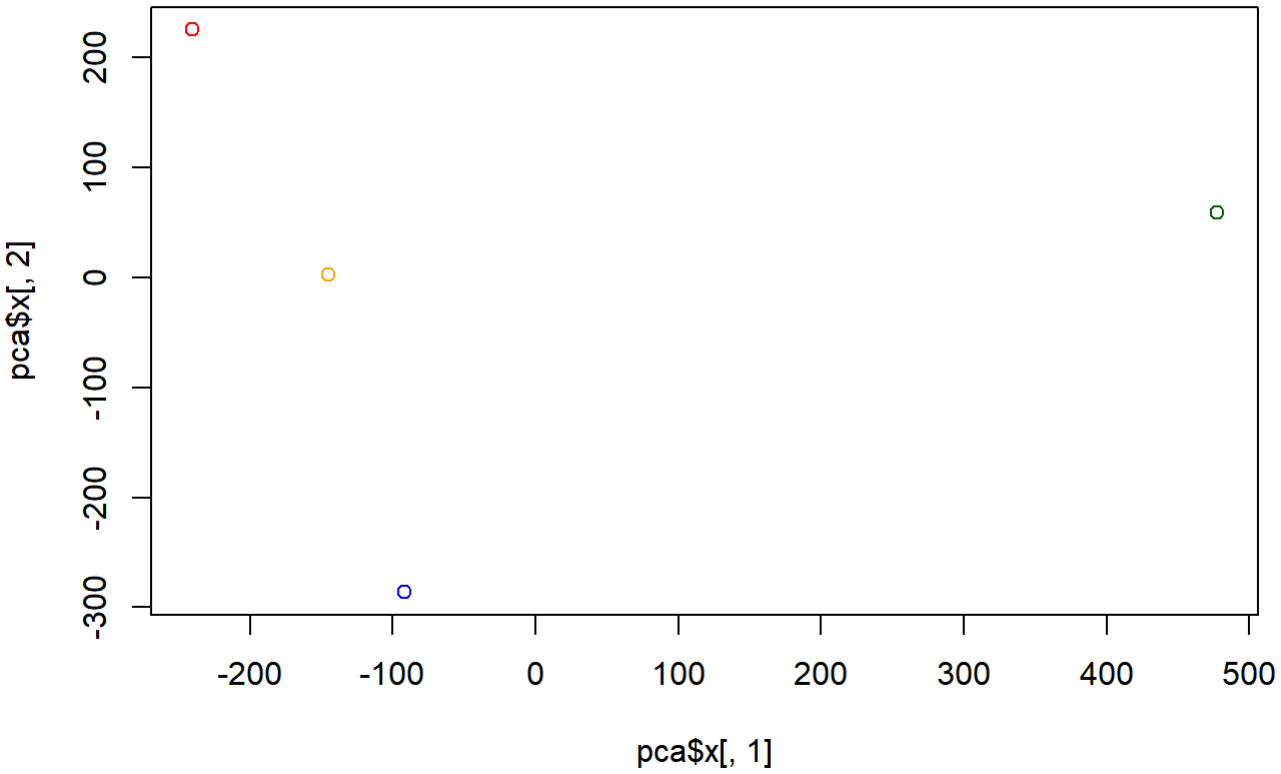
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```
head(pca$x)
```

```
                PC1         PC2         PC3           PC4
England   -144.99315    2.532999 -105.768945  2.842865e-14
Wales     -240.52915  224.646925   56.475555  7.804382e-13
Scotland   -91.86934 -286.081786   44.415495 -9.614462e-13
N.Ireland  477.39164   58.901862    4.877895  1.448078e-13
```

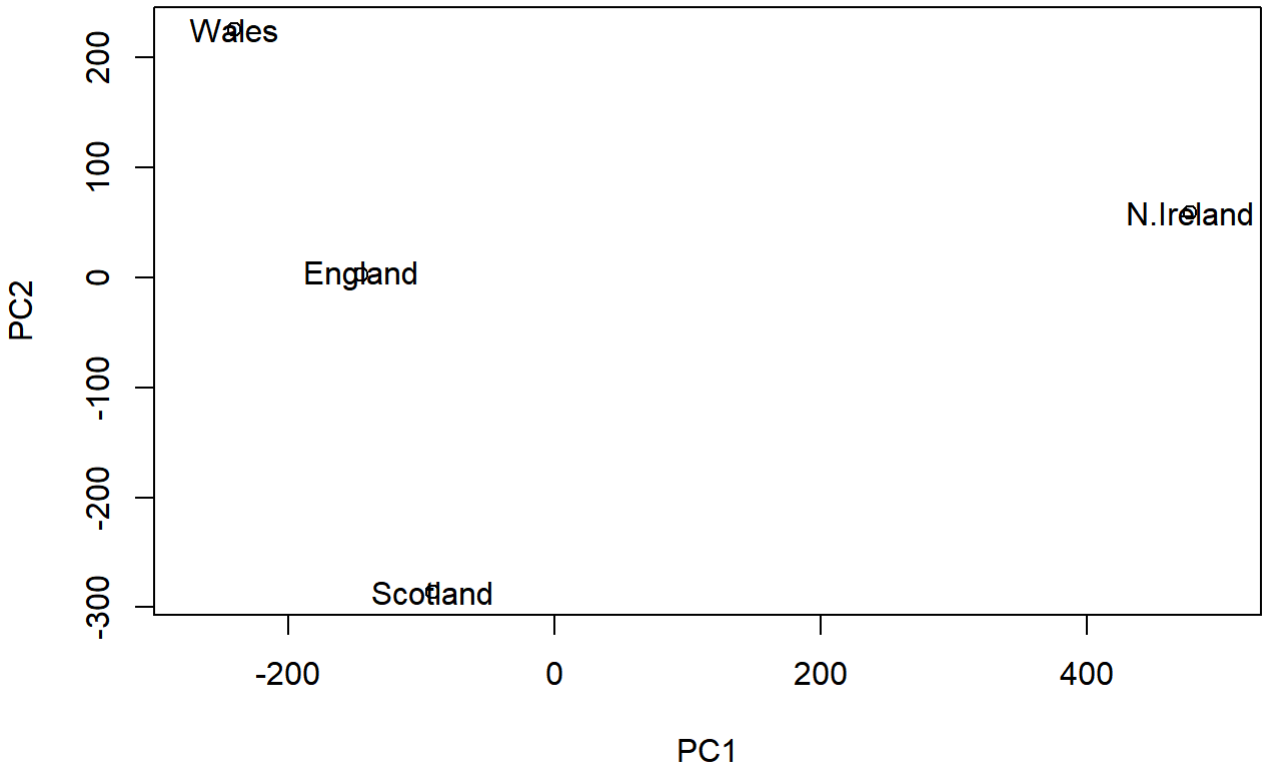
Let's plot our main result: >Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
plot(pca$x[,1],pca$x[,2],col=c("orange","red","blue","darkgreen"))
```



> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

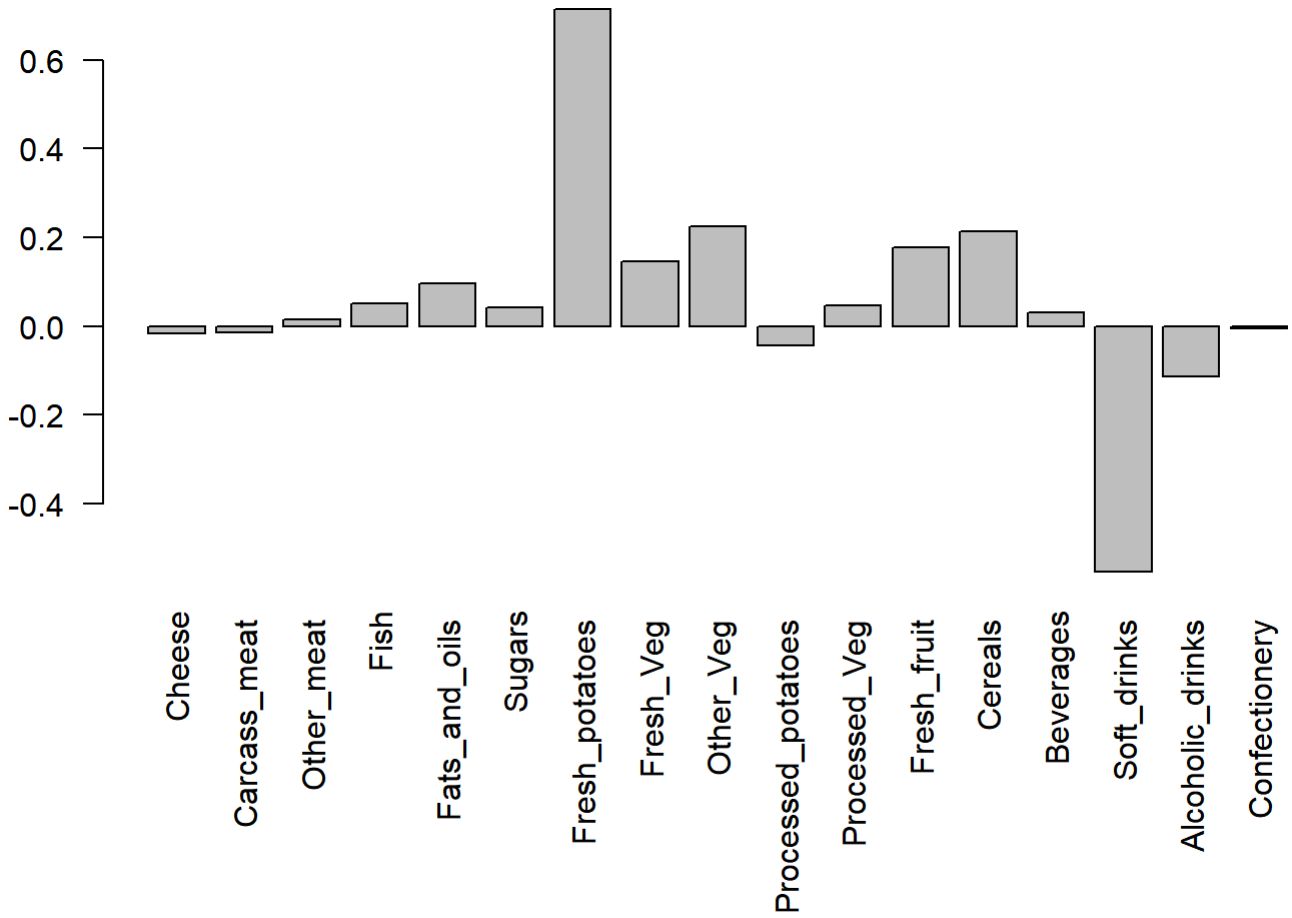
```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x))
```



New axis basically

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

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



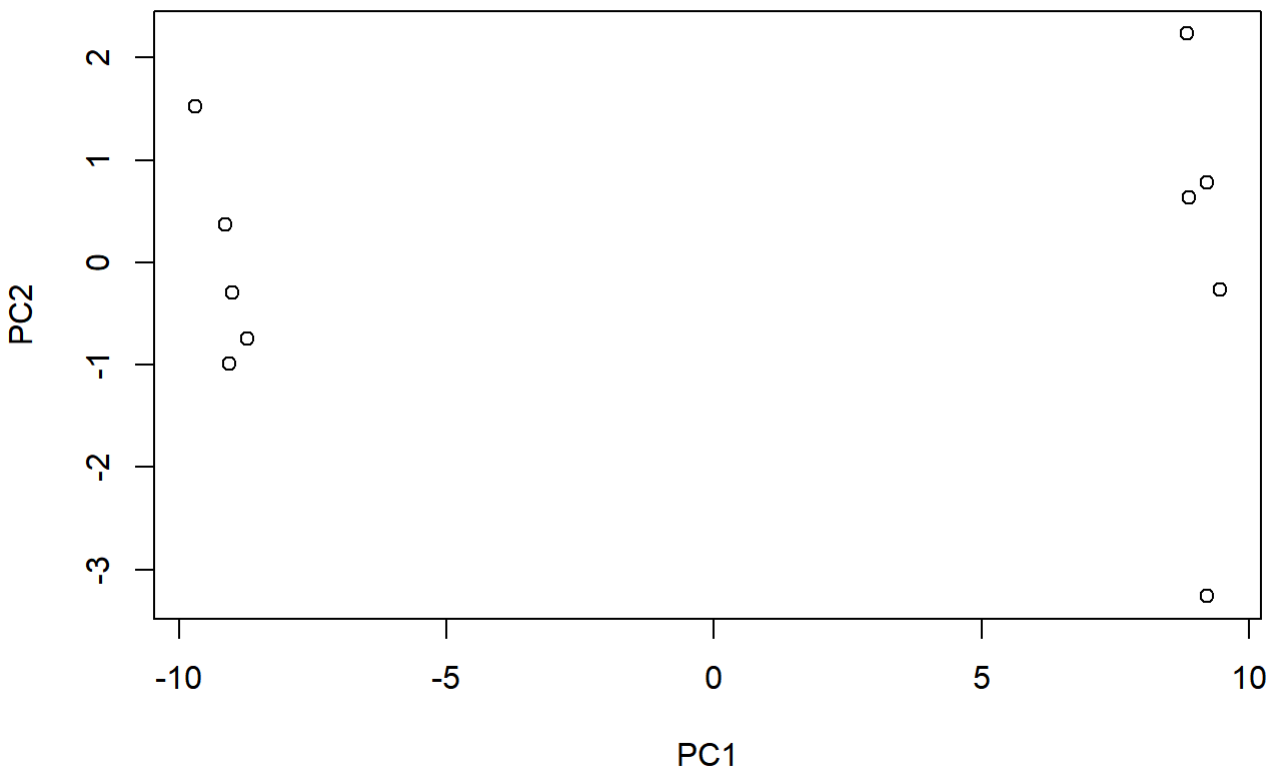
Fresh potatoes and soft_drinks features predominately and PC2 mainly tell us the information about the second dimension (2nd axis).

Part 2 :

```
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
```

```
pca <- prcomp(t(rna.data), scale=TRUE)
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```


PC2
2
1
0
-1
-2
-3
-10
-5
0
5
10
PC1