Class_08

Anika Bhattacharjya (A15459876)

10/21/2021

First up is clustering methods

#Kmeans clustering

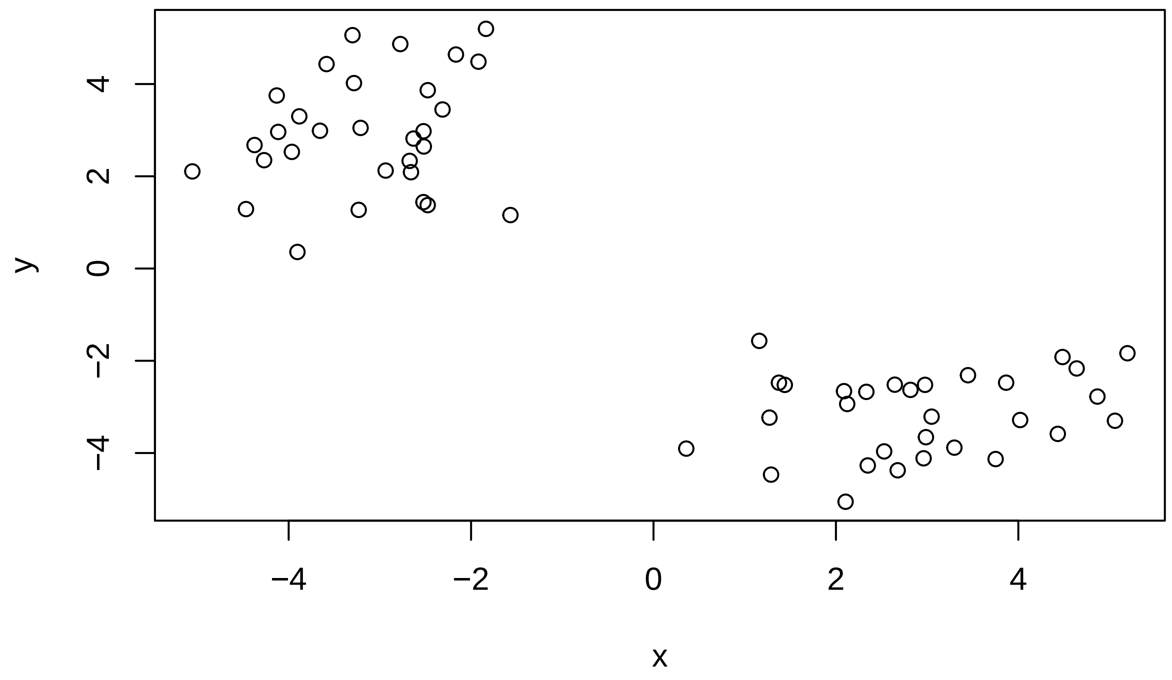
The function in base R to do Kmeans clustering is called 'kmeans()'

First make up some data where we know what the answer should be:

```
tmp <- c(rnorm(30,-3), rnorm(30,3))
x <- cbind(x=tmp, y=rev(tmp))
x
```

```
##               x        y
##  [1,] -3.903877  0.359280
##  [2,] -2.521531  2.976836
##  [3,] -3.284051  4.020068
##  [4,] -3.883268  3.298901
##  [5,] -2.936833  2.123831
##  [6,] -4.130707  3.751103
##  [7,] -1.836739  5.196439
##  [8,] -2.475593  1.375293
##  [9,] -4.114520  2.961445
## [10,] -5.056103  2.106063
## [11,] -3.964889  2.529022
## [12,] -2.631064  2.817329
## [13,] -2.776078  4.867017
## [14,] -3.211672  3.049036
## [15,] -1.918888  4.484413
## [16,] -3.584845  4.432769
## [17,] -3.232274  1.271517
## [18,] -2.312825  3.448025
## [19,] -4.269171  2.349272
## [20,] -2.475074  3.866457
## [21,] -2.658771  2.088574
## [22,] -1.568452  1.159585
## [23,] -3.656608  2.987012
## [24,] -4.374141  2.677371
## [25,] -2.522436  1.439703
## [26,] -2.673494  2.333662
## [27,] -2.518130  2.645605
## [28,] -4.468089  1.289056
## [29,] -2.165905  4.639808
## [30,] -3.300228  5.059465
## [31,]  5.059465 -3.300228
## [32,]  4.639808 -2.165905
## [33,]  1.289056 -4.468089
## [34,]  2.645605 -2.518130
## [35,]  2.333662 -2.673494
## [36,]  1.439703 -2.522436
## [37,]  2.677371 -4.374141
## [38,]  2.987012 -3.656608
## [39,]  1.159585 -1.568452
## [40,]  2.088574 -2.658771
## [41,]  3.866457 -2.475074
## [42,]  2.349272 -4.269171
## [43,]  3.448025 -2.312825
## [44,]  1.271517 -3.232274
## [45,]  4.432769 -3.584845
## [46,]  4.484413 -1.918888
## [47,]  3.049036 -3.211672
## [48,]  4.867017 -2.776078
## [49,]  2.817329 -2.631064
## [50,]  2.529022 -3.964889
## [51,]  2.106063 -5.056103
## [52,]  2.961445 -4.114520
## [53,]  1.375293 -2.475593
## [54,]  5.196439 -1.836739
## [55,]  3.751103 -4.130707
## [56,]  2.123831 -2.936833
## [57,]  3.298901 -3.883268
## [58,]  4.020068 -3.284051
## [59,]  2.976836 -2.521531
## [60,]  0.359280 -3.903877
```

```r
plot(x)
```



Q. Can we use kmeans to cluster this data setting k 2 and nstart to 20

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

```
## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##           x         y
## 1 -3.147542  2.920132
## 2  2.920132 -3.147542
##
## 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 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
##
## Within cluster sum of squares by cluster:
## [1] 69.76774 69.76774
##  (between_SS / total_SS =  88.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Q. How many points are in each cluster?

```r
km$size
```

```
## [1] 30 30
```

Q. What ‘component’ of your result object details cluster assignment/membership?

```r
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 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
```

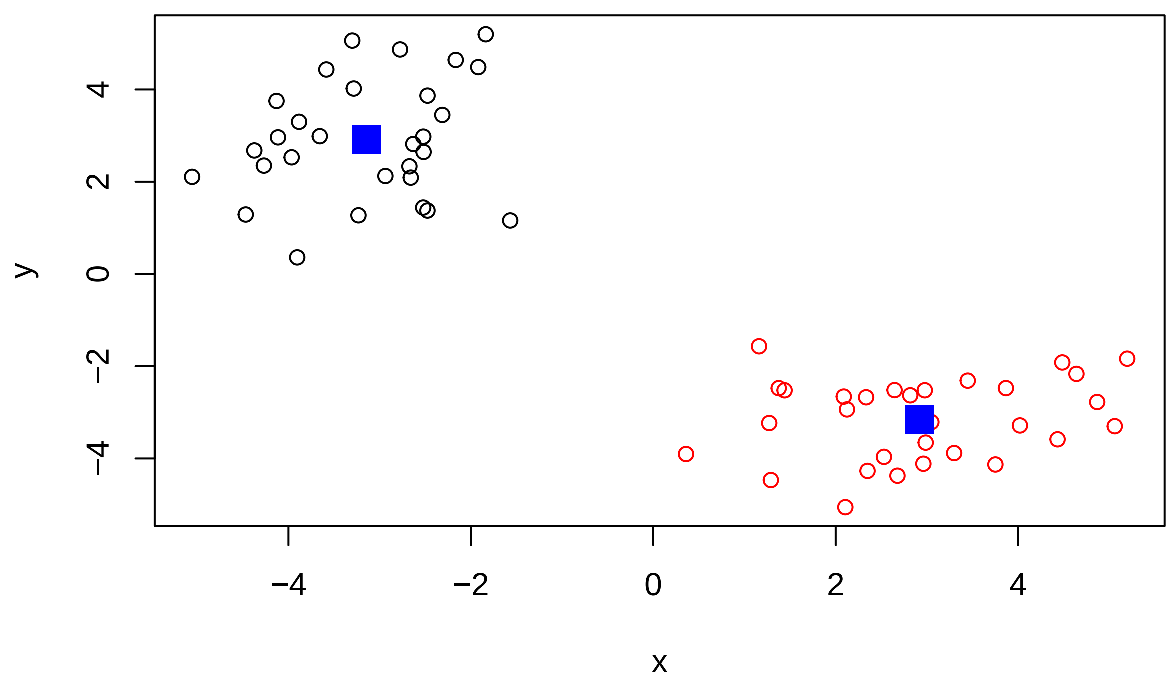
What ‘component’ of your result object details cluster centers?

```r
km$centers
```

```
##           x         y
## 1 -3.147542  2.920132
## 2  2.920132 -3.147542
```

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

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



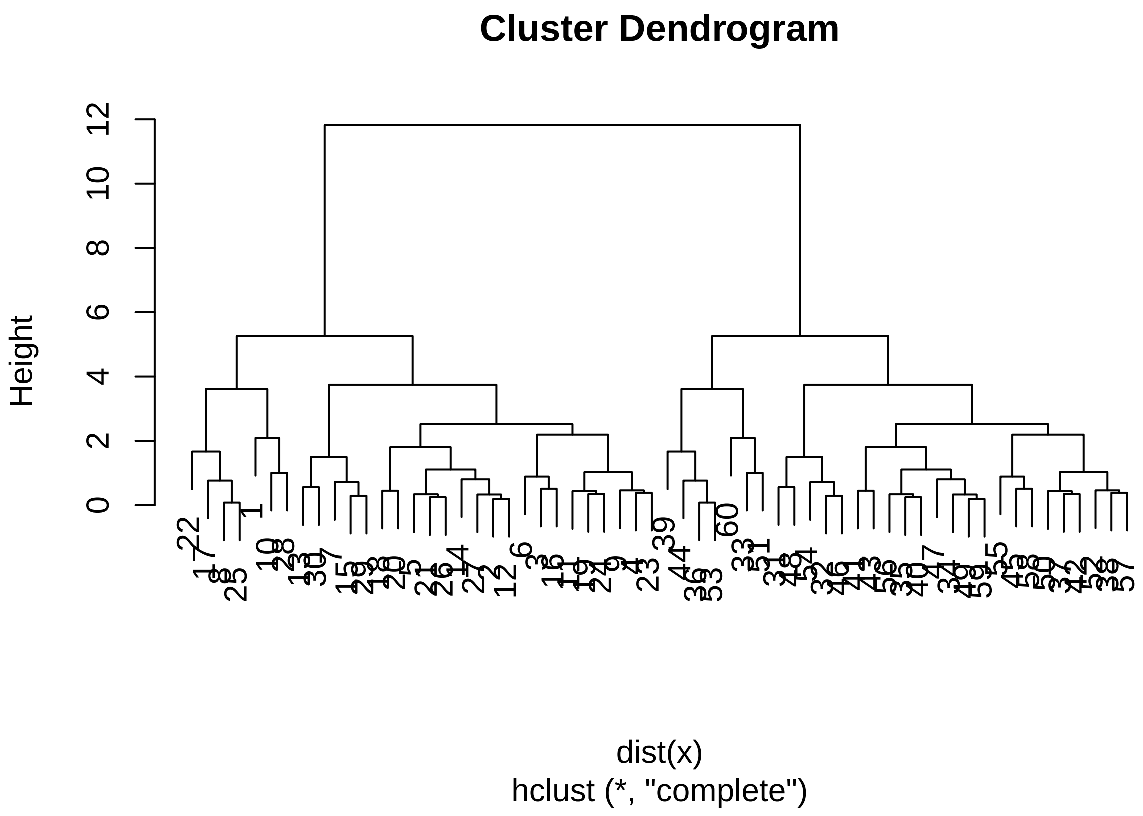
Analyze same data with hclust()

```r
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 result object

```r
plot(hc)
```



To get our cluster membership vector we need to do a little more work. We have to "cut" the tree where we think it makes sense. For this we use `cutree()` function.

```r
cutree(hc, h = 6)
```

```
##  [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
```

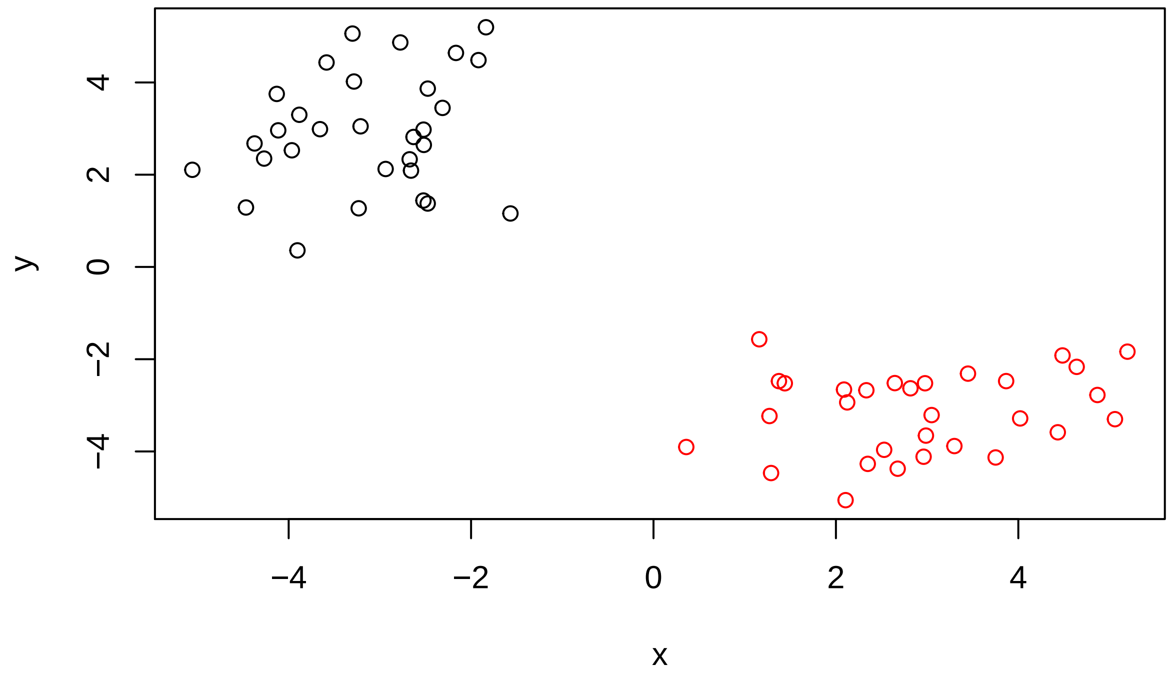
You can also call `cutree()` setting k = numner of grps/clusters you want.

```r
cutree(hc, k=2)
```

```
##  [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
```

Make our results plot

```r
grps <- cutree(hc,k=2)
plot(x, col=grps)
```



#PCA of UK food data

import data

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

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

```r
dim(x)
```

```
## [1] 17  5
```

Checking your data

Preview first 6 rows

```
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
## 7      Fresh_potatoes      720   874      566      1033
## 8           Fresh_Veg      253   265      171       143
## 9           Other_Veg      488   570      418       355
## 10 Processed_potatoes      198   203      220       187
## 11      Processed_Veg      360   365      337       334
## 12        Fresh_fruit     1102  1137      957       674
## 13            Cereals     1472  1582     1462      1494
## 14           Beverages      57    73       53        47
## 15         Soft_drinks    1374  1256     1572      1506
## 16    Alcoholic_drinks     375   475      458       135
## 17      Confectionery       54    64       62        41
```

Row-names are incorrectly set as the first column of our x data fram. Use minus indexing as one option.

```
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
```

There is a problem here because if we keep inputing the function it takes a column away each time.

Another method:

```
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
dim(x)
```

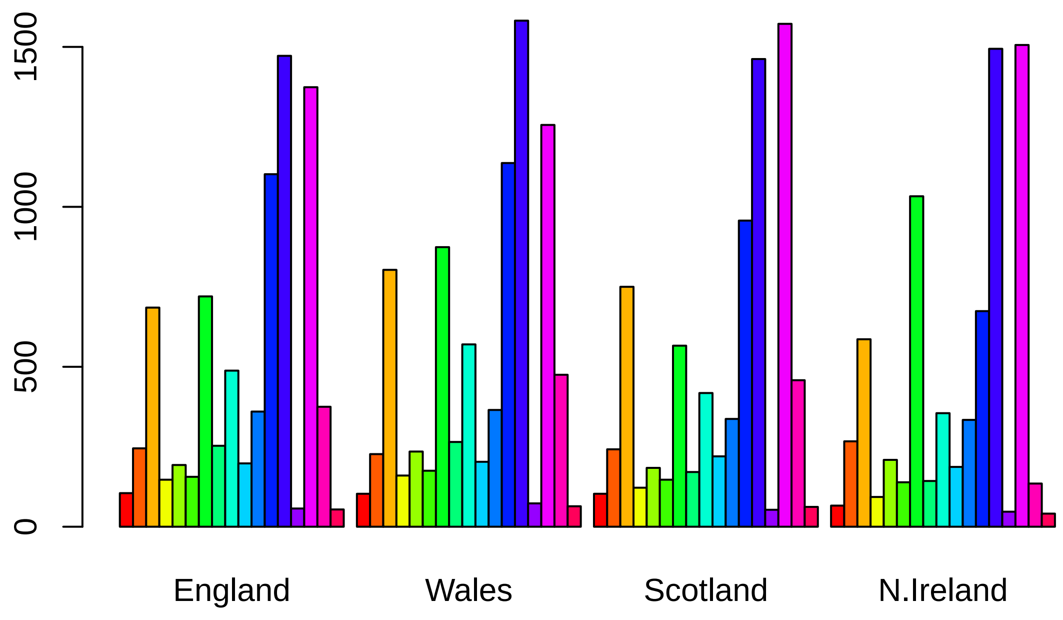
```
## [1] 17  4
```

Now we have correct dimensions > 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 option is the better option because it doesn't keep taking away columns.

Plot the data

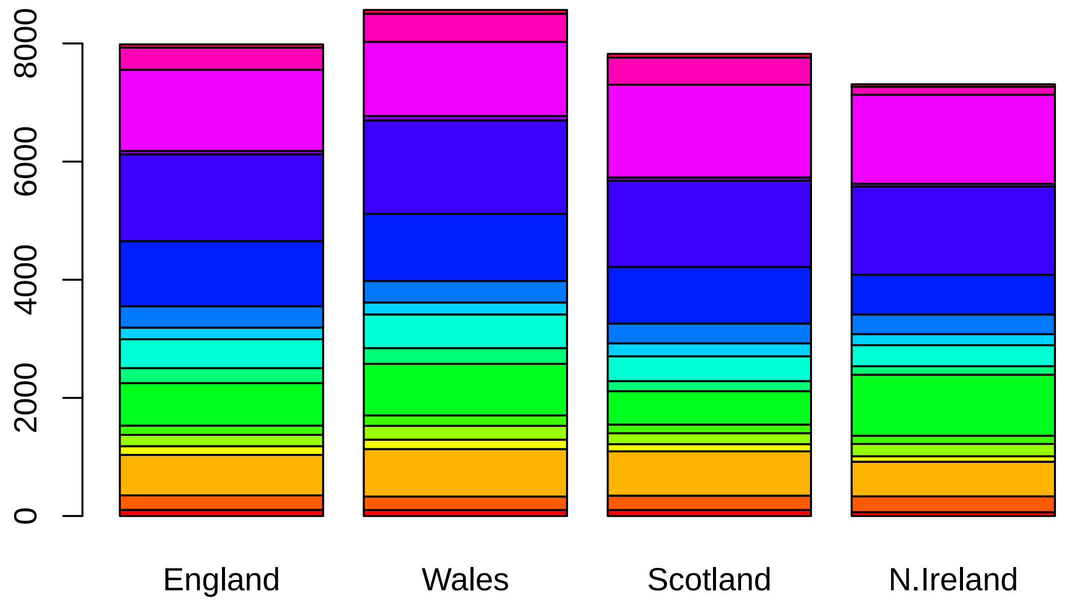
```r
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

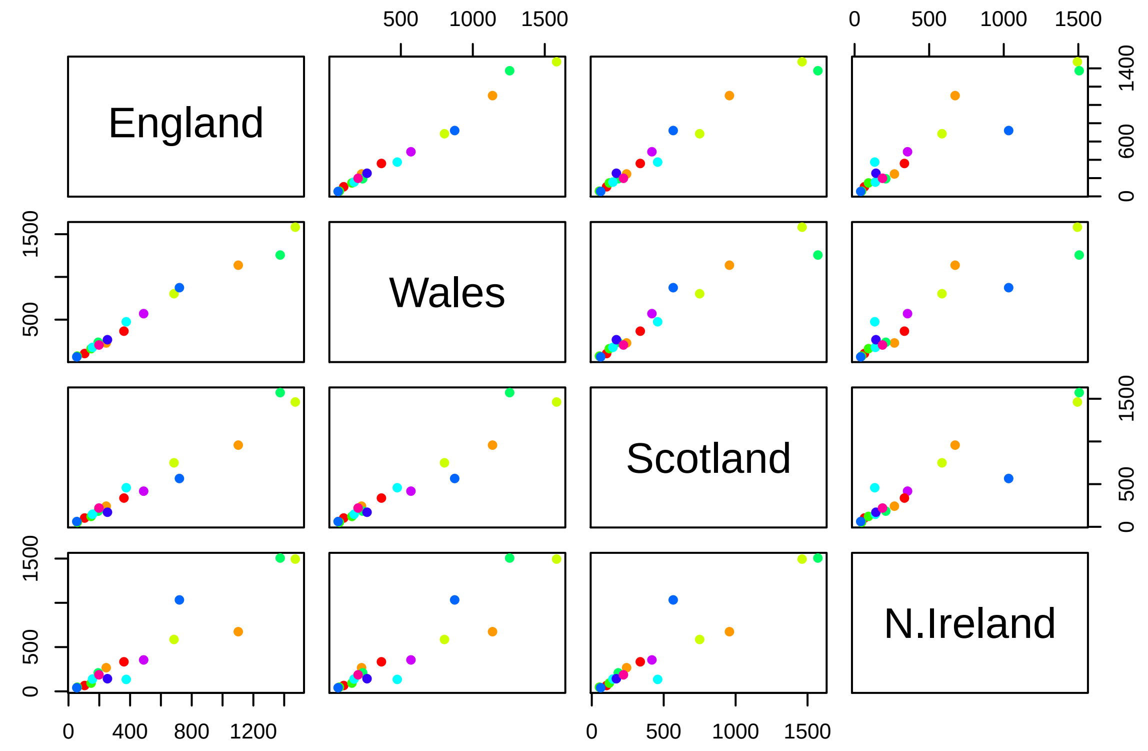
Set beside to false

```r
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?

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



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

Can’t really tell the difference at this point the way the data is presented.

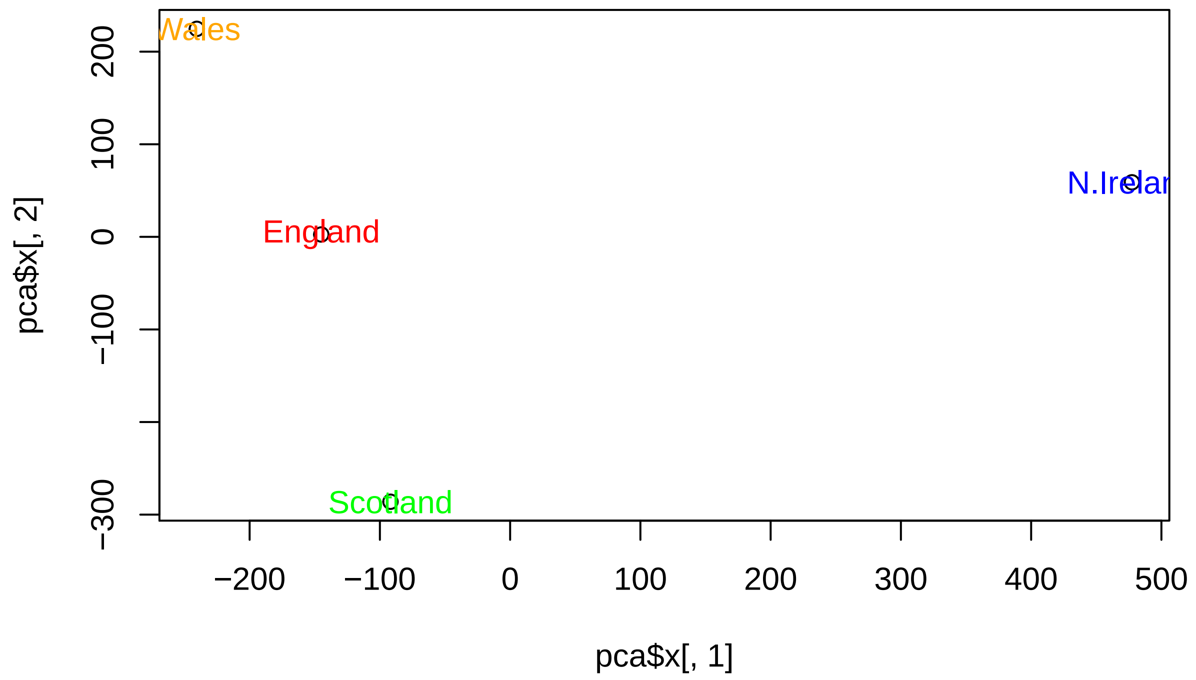
Use PCA Funtion

```r
# Use the prcomp() PCA function
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
```

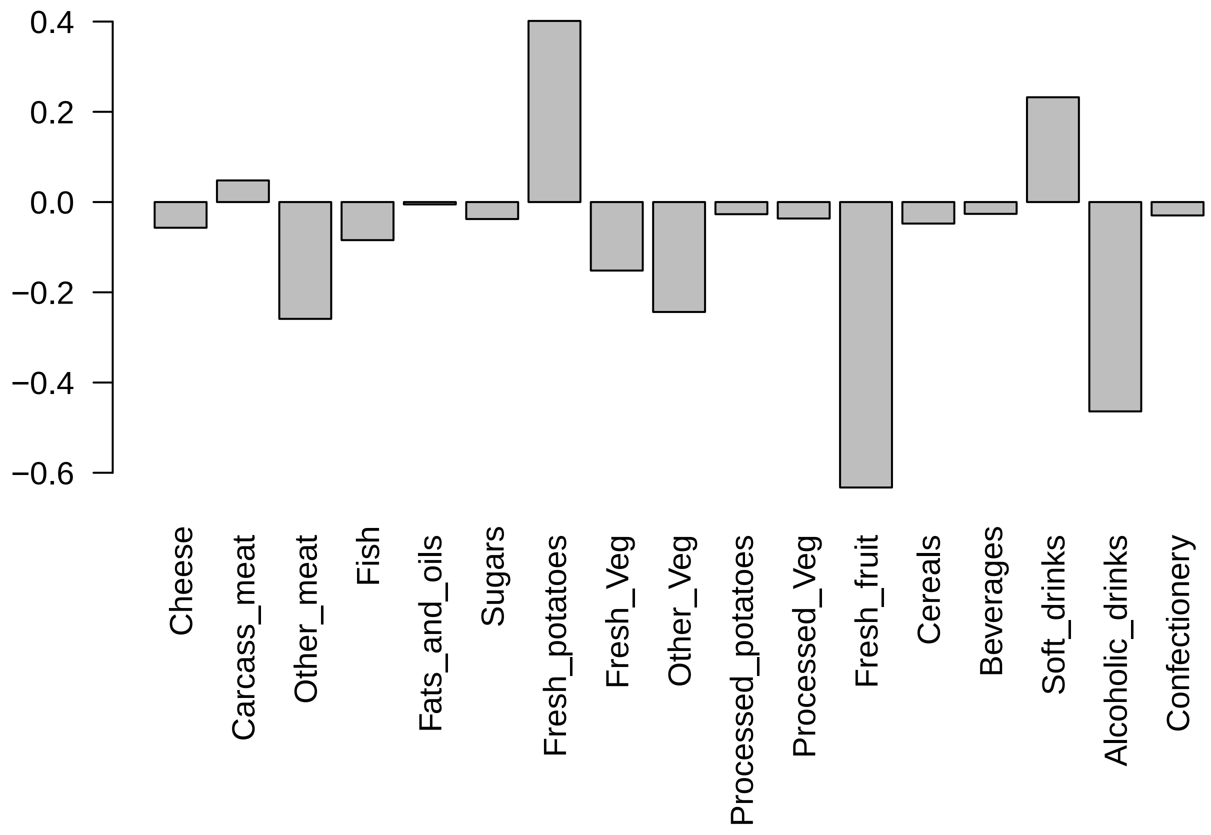
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```r
#Plot PC1 vs PC2
colors <- c("red", "Orange", "Green", "Blue")
plot(pca$x[,1], pca$x[,2])
text(pca$x[,1], pca$x[,2], colnames(x), col= colors)
```



Variable Loadings!

```r
# Focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot(pca$rotation[,1], las=2)
```


0.4
0.2
0.0
−0.2
−0.4
−0.6
Cheese
Carcass_meat
Other_meat
Fish
Fats_and_oils
Sugars
Fresh_potatoes
Fresh_Veg
Other_Veg
Processed_potatoes
Processed_Veg
Fresh_fruit
Cereals
Beverages
Soft_drinks
Alcoholic_drinks
Confectionery