Class 7 Lab: Machine Learning I

Pamelina Lo (PID:A16735368)

Today we are going to learn how to apply different machine learning methods, beginning with clustering:

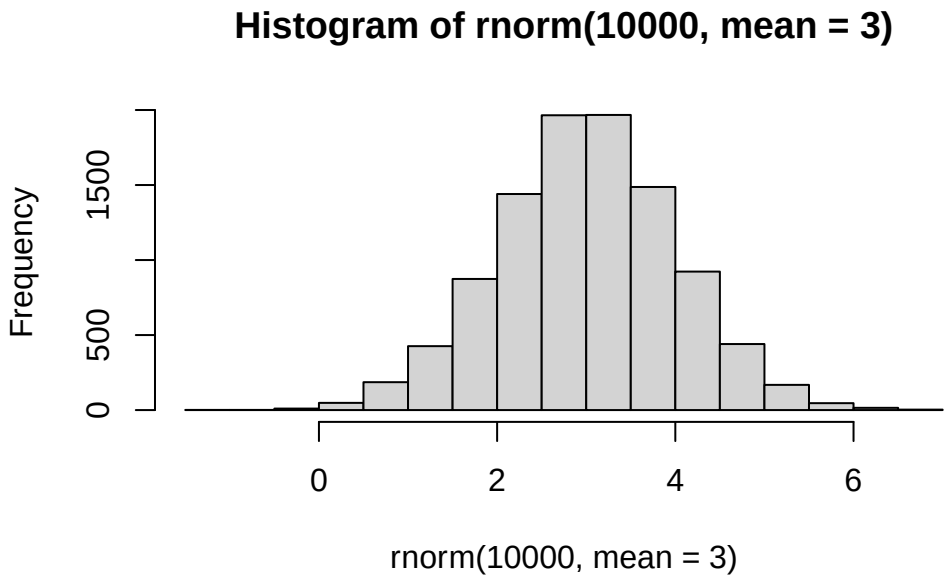
The goal here is to find groups/ clusters in you input data.

First, I will make up some data with clear groups. For this I will use the `rnorm()` function:

```
rnorm(10)
```

```
 [1]  0.9139520  0.7961771 -0.1287992  0.6750854  1.2749951  0.5606630
 [7]  0.9094944  1.9258036 -0.1365476  0.7512282
```

```
hist(rnorm(10000, mean = 3))
```

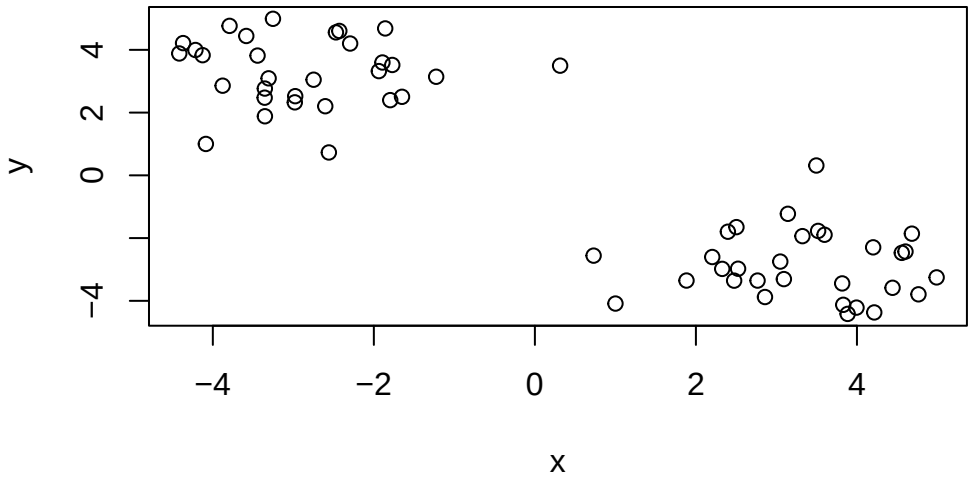


You can put this into a vector using 'c()' function.

```
n=30
x <- (c(rnorm(n, mean =-3), rnorm(n, mean =3)))
y <- rev(x)
z <- cbind(x,y)

head(z)
```

```
             x        y
[1,] -1.937712 3.322964
[2,] -1.893523 3.597576
[3,] -1.859013 4.682415
[4,] -2.748323 3.047879
[5,] -4.126843 3.829820
[6,] -4.416637 3.884638
```

```
plot(z)
```



Class Discussion:

kmeans()

Use the `kmeans()` dunction k to 2 and nstart=20

Inspect/print results

> Q. How many points are in each cluster? Q. What 'components' of your results - cluster size? - cluster alignment/membership? - cluster center?

```
km <- kmeans(z, centers =2)
km
```

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

Cluster means:
          x         y
1 -2.847472  3.295444
2  3.295444 -2.847472

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 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

Within cluster sum of squares by cluster:
[1] 67.75407 67.75407
 (between_SS / total_SS =  89.3 %)

Available components:

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

Results in kmeans object `km`

```
attributes(km)
```

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

```
$class
[1] "kmeans"
```

cluster size?

```
km$size
```

```
[1] 30 30
```

cluster assignment/memebership?

```
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 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
```

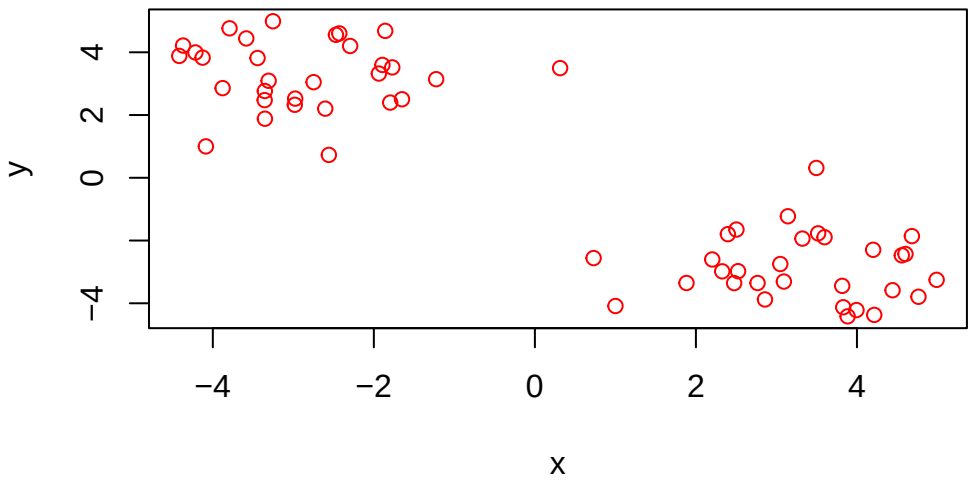
cluster center?

```
km$centers
```

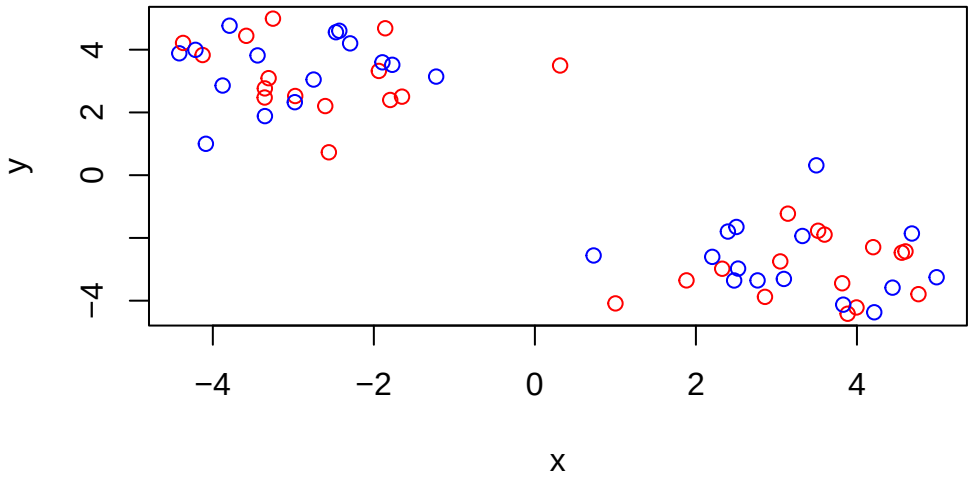
```
          x         y
1 -2.847472  3.295444
2  3.295444 -2.847472
```

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

```
plot(z,col="red")
```

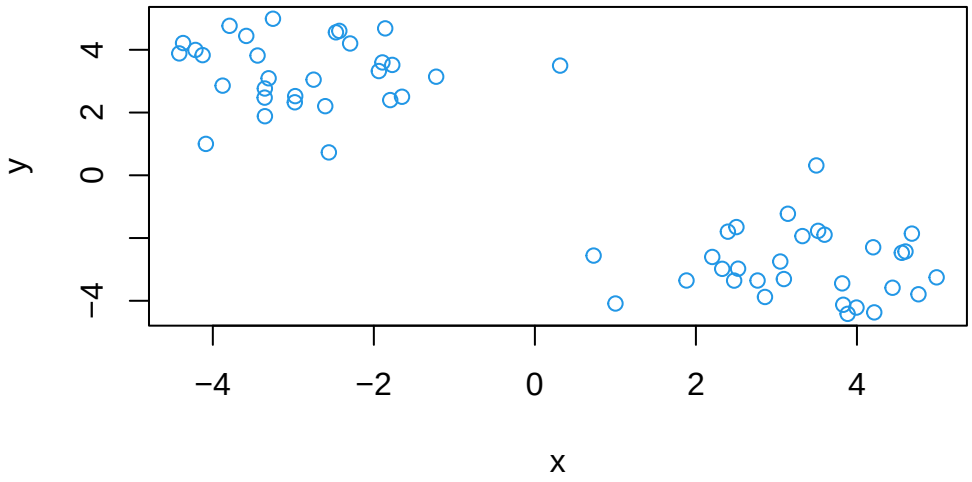



```
plot(z,col= c("red", "blue"))
```

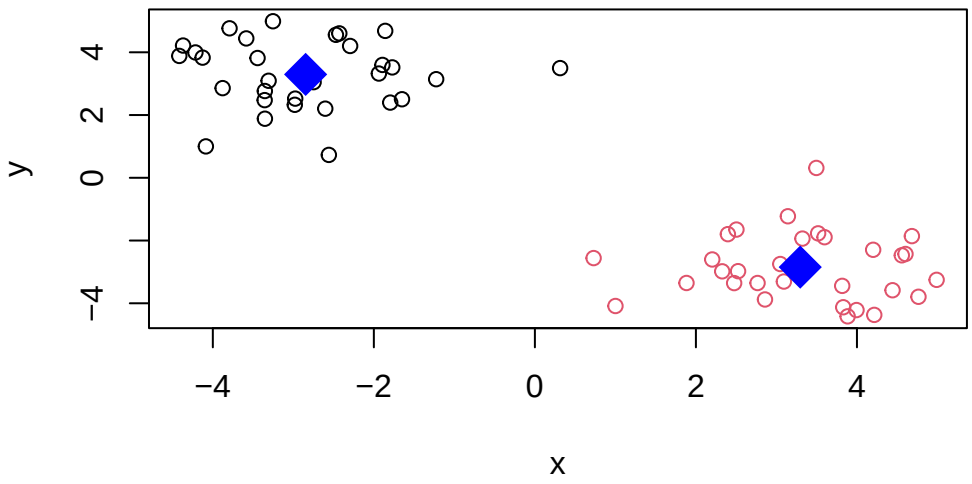


R will re-cycle the shorter color vector to be the same length as the longer (number of data points) in it.

```r
plot(z, col=100)
```




```r
plot(z, col=km$cluster)
points(km$centers, col="blue", pch=18, cex=3)
```



Q. Can you turn kmeans and ask for 4 clusters please and plot the results like we have done above?

```
pm <- kmeans(z, centers =4)
pm
```

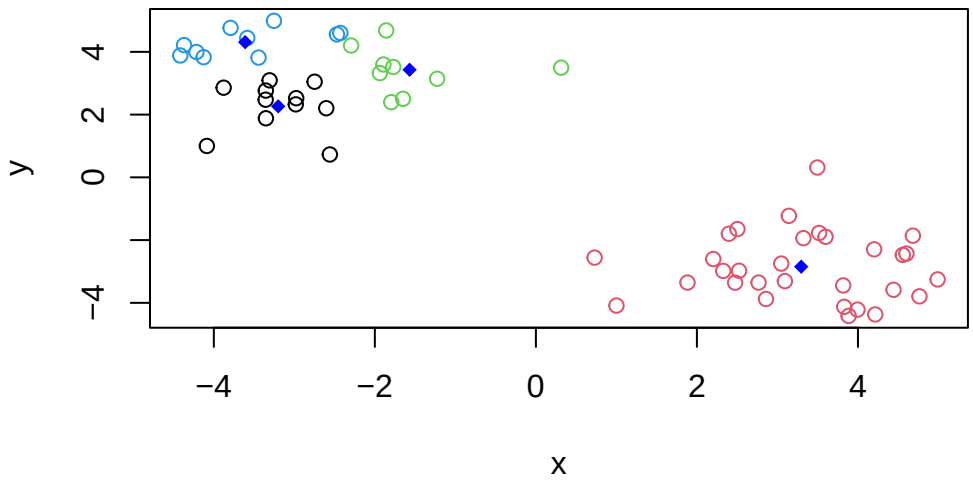
```
K-means clustering with 4 clusters of sizes 11, 30, 9, 10

Cluster means:
          x         y
1 -3.200405  2.264125
2  3.295444 -2.847472
3 -1.568416  3.428962
4 -3.610395  4.309729

Clustering vector:
 [1] 3 3 3 1 4 4 1 1 1 3 1 1 1 4 4 1 1 3 3 4 4 3 3 4 1 4 4 4 3 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]  8.517594 67.754073  8.835778  6.338160
 (between_SS / total_SS =  92.8 %)
```

```
Available components:

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

```
attributes(pm)
```

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

$class
[1] "kmeans"
```

```
plot(z, col=pm$cluster)
points(pm$centers, col="blue", pch=18, cex=1)
```



You can add more clusters by replacing the number of centers with `kmeans()` functions. See above.

Hierarchical Clustering

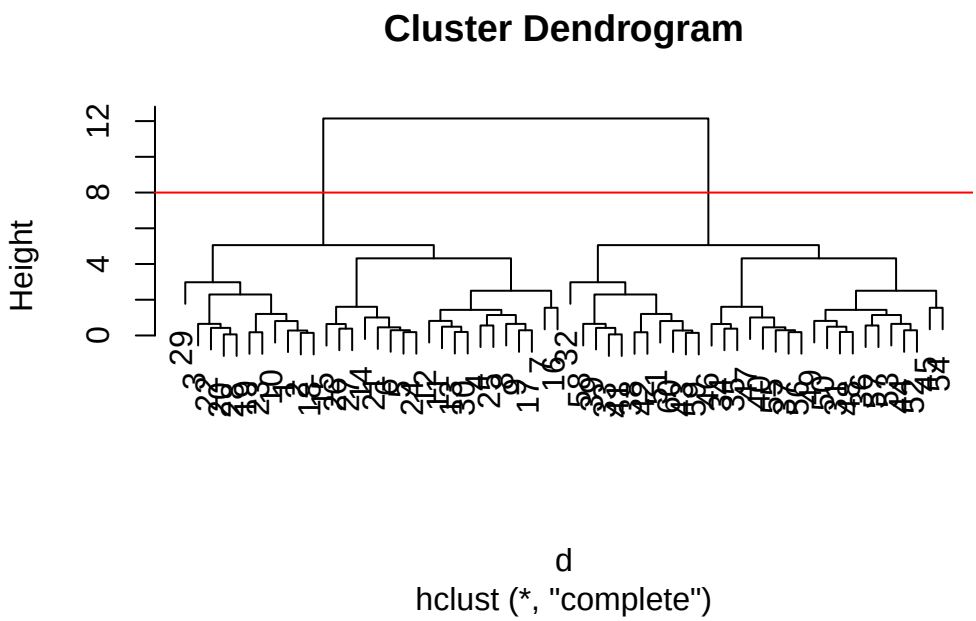
Let's take our some made-up data **z** and see how hclust works.

First we need a distance matrix for our data to be clustered.

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

```
Call:
hclust(d = d)

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

```
plot(hc)

#To add a line on your plot
abline(h=8, col = "red")
```



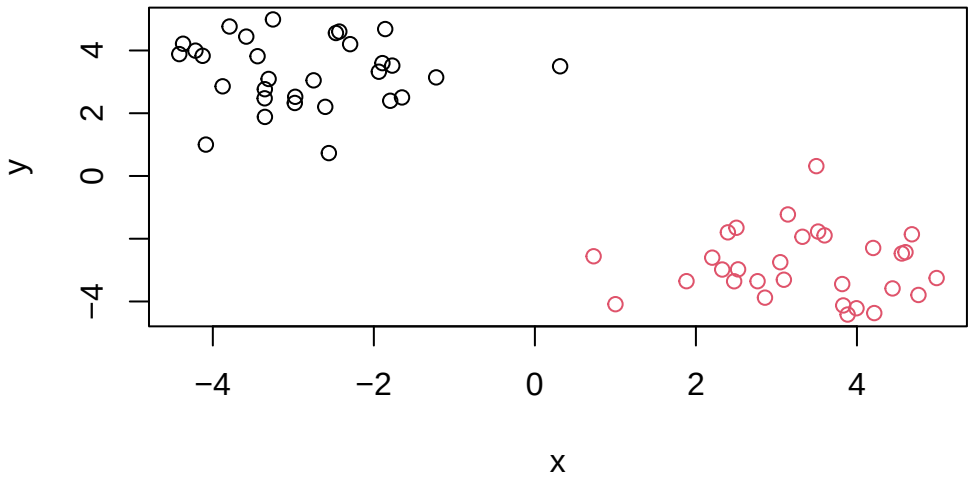
I can get my cluster membership vectory by "cutting the tree" with the `cutree()` function like so:

```
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 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
```

Can you plot `z` colored by our hclust results:

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



PCA of UK food data

Read data from the UK food consumption in different parts of the UK.

```
url <- "https://tinyurl.com/UK-foods"
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
```

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

```
dim(x)
```

```
[1] 17  4
```

```
ncol(x)
```

```
[1] 4
```

```
nrow(x)
```

```
[1] 17
```

Checking your data

```
## 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
```

```
dim(x)
```

```
[1] 17  3
```

```
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
```

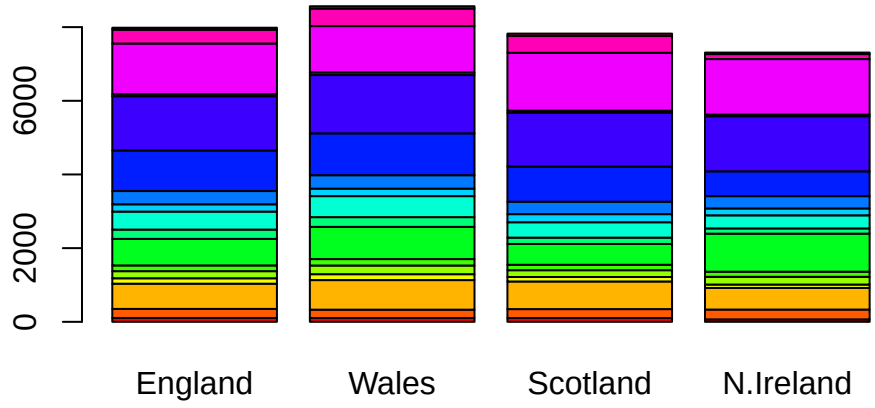
Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances? The 'row.names()' function would be the best approach to solve the problem because the output appears more organized and mucn more cleaner. Yes, this approach is more robust than the other.

```
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
```

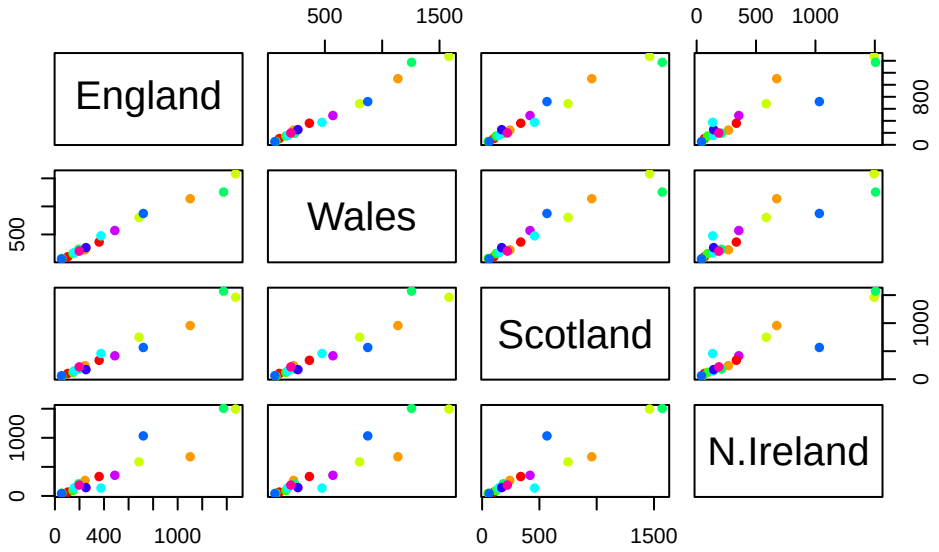
Q3: Changing what optional argument in the above barplot() function results in the following plot? You change the `beside =` function from TRUE (T) to FALSE (F) in the barplot() function results in the following plot.

```
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?

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



Its hard to see structure and trends and even this small data set. How will we ever do this when wh have big datta sets with 1,000s or 10s of thousands of things we are measuring.

Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set? The main difference between N. Ireland and the other countries of the UK is that N. Ireland has a some scattered data/points (which falls out of the diagonal line of data) compared to the other UK countries which suggests that the data has some differences than the other data.

PCA to the Rescue

Let's see how PCS deals with thsi dataset. So main functions in base R to do PCA is called `prcomp()`.

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

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 3.176e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

Let's see what is in inside this `pca` object that we created from running `prcomp()`.

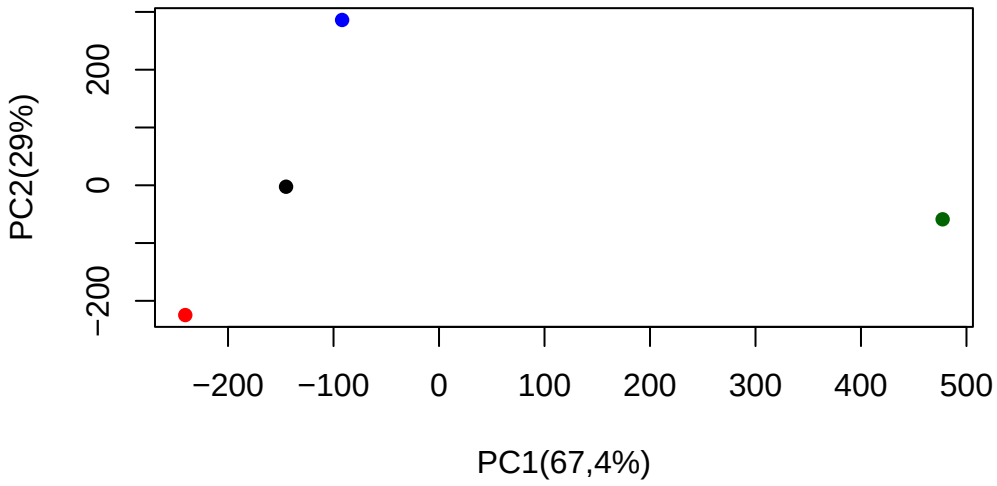
```
attributes(pca)
```

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

$class
[1] "prcomp"
```

```
pca$x
```

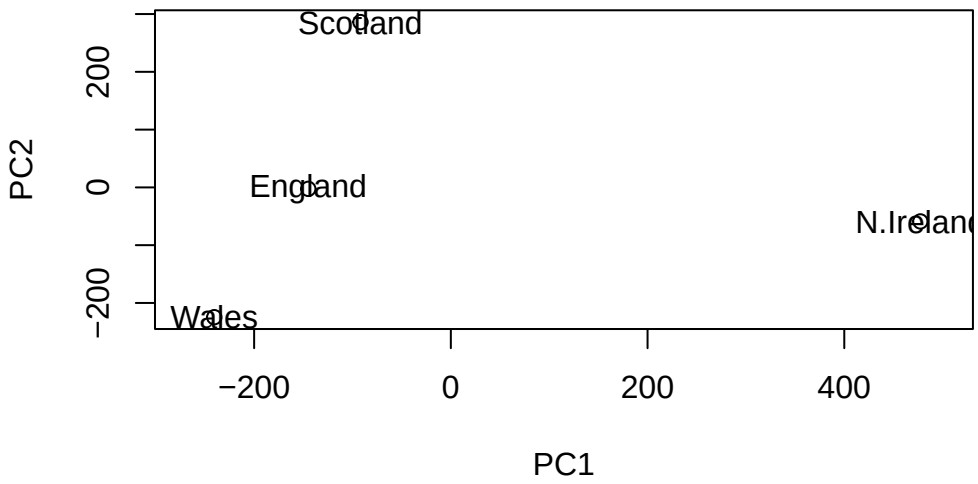
```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -4.894696e-14
Wales     -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland   -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland  477.39164  -58.901862  -4.877895  2.321303e-13
```

```
plot(pca$x[,1], pca$x[,2],
     col=c("black", "red","blue", "darkgreen"), pch=16,
     xlab = "PC1(67,4%)" , ylab = "PC2(29%)" )
```



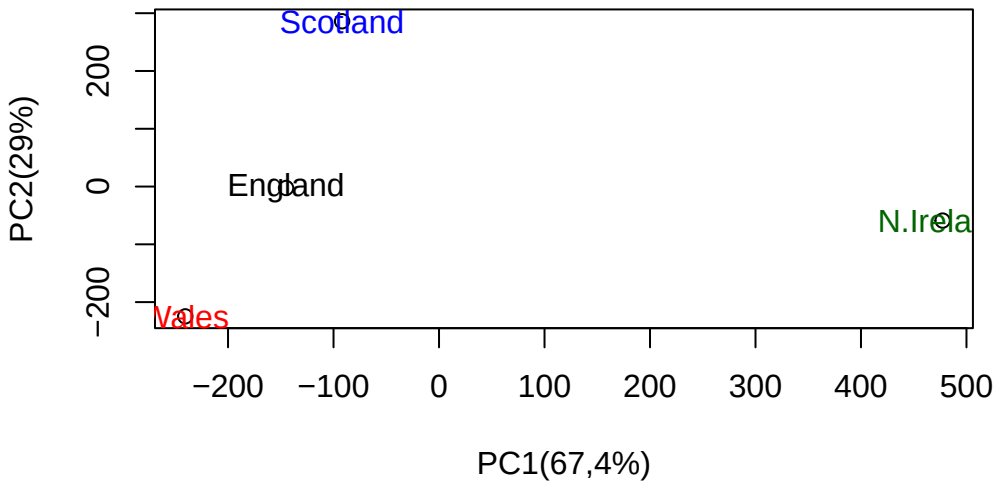
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

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



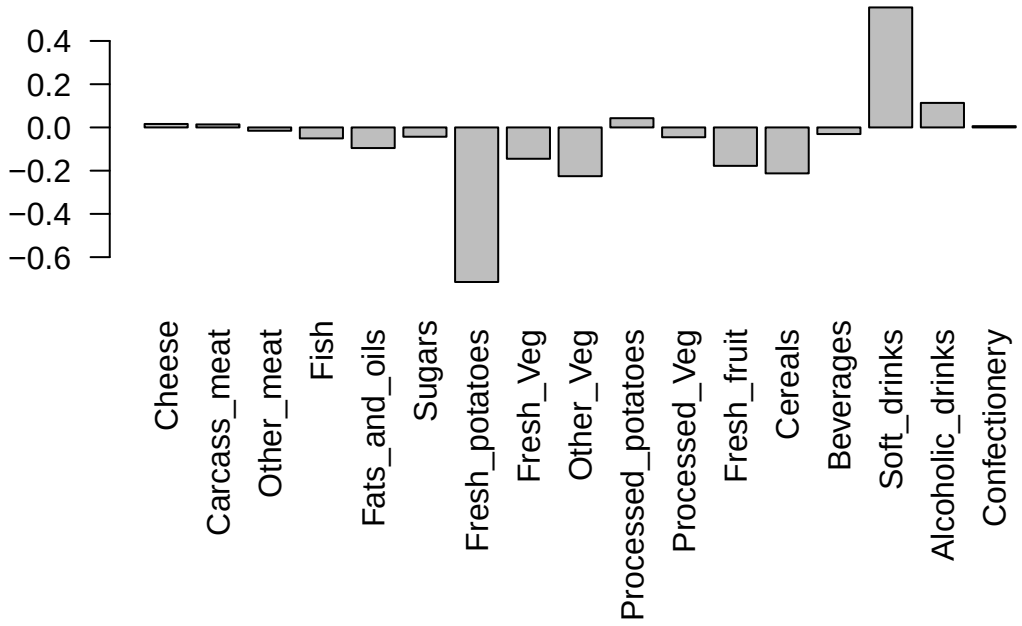
Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
plot(pca$x[,1], pca$x[,2],
     xlab = "PC1(67,4%)" , ylab = "PC2(29%)")
     text(pca$x[,1], pca$x[,2], colnames(x), col=c("black", "red","blue", "darkgreen", pch=1
```



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

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



The two food groups Fresh_potatoes and Soft_drinks. PC2 tells us about the food of the best possible loading scores in all UK countries in which the best possible loading score for fresh potatoes is Scotland and N. Ireland, England, and Wales have the best loading score in soft drinks.

PCA of RNA-seq data

```
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
```

Q10. How many genes and samples are in this data?

```
dim(rna.data)
```

```
[1] 100  10
```

```
ncol(rna.data)
```

```
[1] 10
```

```
nrow(rna.data)
```

```
[1] 100
```