Class 7: Clustering and PCA

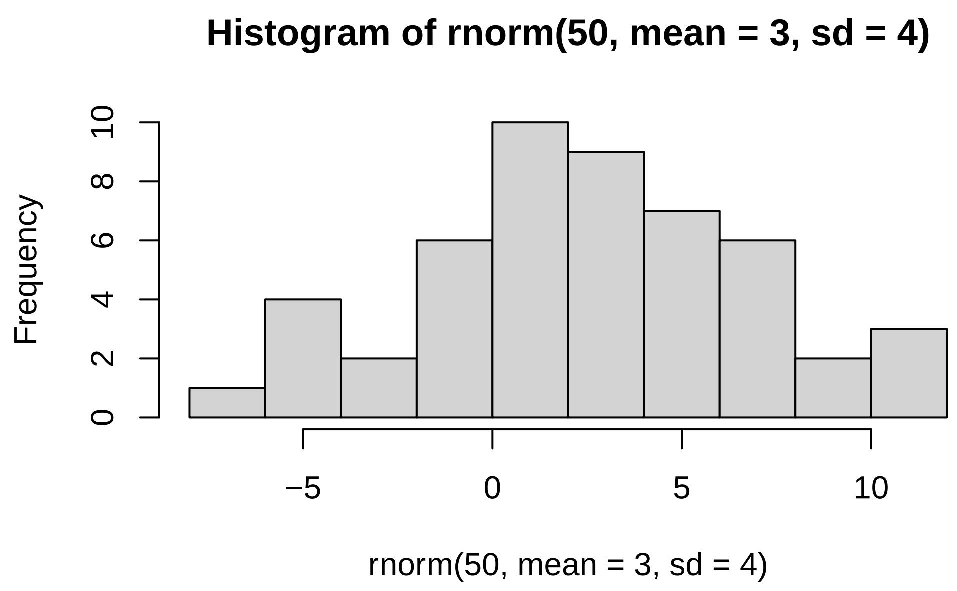
Angela Bartolo PID: A19532451

Clustering

First let's make up some data to cluster so we can get a feel for these methods and how to work with them.

We can use the 'rnorm()" function to get random numbers from a normal distribution around a given 'mean'.

```r
hist(rnorm(50, mean=3, sd=4))
```



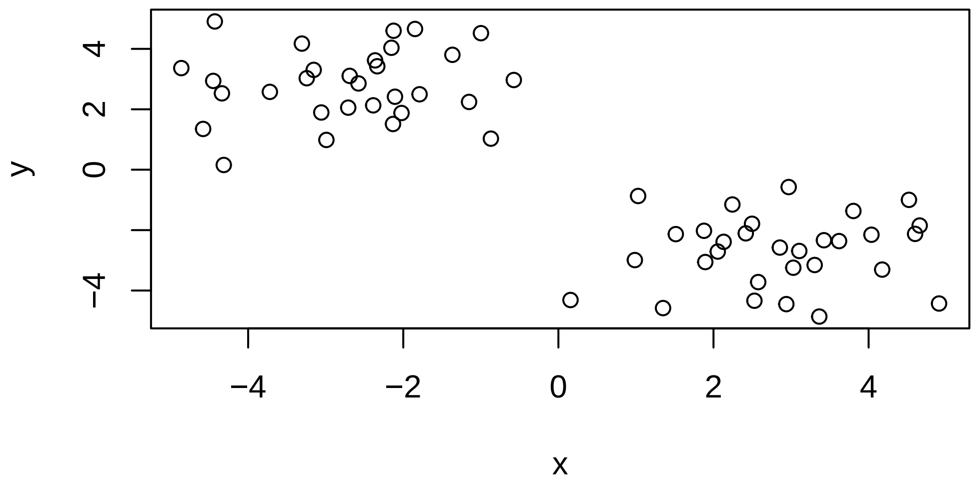
Let's get 30 points with a mean of 3 and another 30 with a mean of -3.

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

```
 [1]  0.1562708  1.3492176  4.0365023  4.9082740  2.4160575  3.3045219
 [7]  1.8771820  2.9692954  3.4237034  2.1307541  4.1750440  0.9858158
[13]  3.8037011  2.5275276  3.1060233  1.5136745  2.2434282  1.8938065
[19]  2.8555714  2.0548467  2.4966673  3.3640423  2.9389497  1.0281483
[25]  4.6585756  4.5202011  2.5766574  4.5994736  3.0289726  3.6193110
[31] -2.3635314 -3.2447670 -2.1248421 -3.7202260 -0.9984265 -1.8487021
[37] -0.8704044 -4.4505584 -4.8619423 -1.7900690 -2.7097458 -2.5776828
[43] -3.0571633 -1.1516786 -2.1324700 -2.6908904 -4.3382307 -1.3665440
[49] -2.9914406 -3.3071907 -2.3878032 -2.3353659 -0.5748604 -2.0225735
[55] -3.1549007 -2.1071144 -4.4300802 -2.1527703 -4.5810360 -4.3145363
```

Put these two together:

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



K-means clustering.

Very popular clustering method that we can use with the 'kmeans()' function in base R.

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

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

Cluster means:
          x         y
1  2.818741 -2.688585
2 -2.688585  2.818741

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] 81.16257 81.16257
 (between_SS / total_SS =  84.9 %)

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 size?

```r
km$size
```

```
[1] 30 30
```

- Cluster assignment/membership?

```
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
```

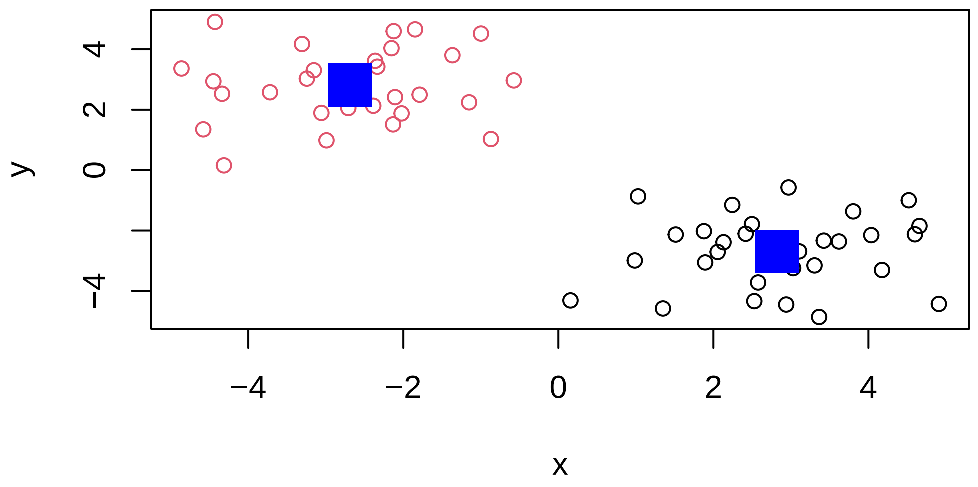
- Cluster center?

```
km$centers
```

```
          x         y
1  2.818741 -2.688585
2 -2.688585  2.818741
```

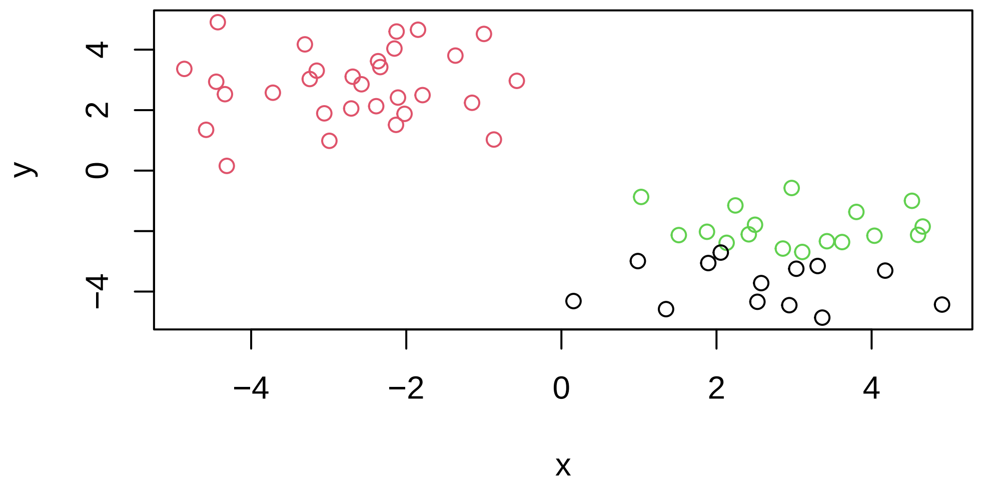
Let's plot

```
plot(x, col=km$cluster)
points(km$centers, col= "blue", pch=15, cex =3)
```



Q Let's cluster into 3 groups or some 'x' data and make a plot.

```
km2 <- kmeans(x, centers=3)
plot(x, col=km2$cluster)
```



Hierarchical Clustering

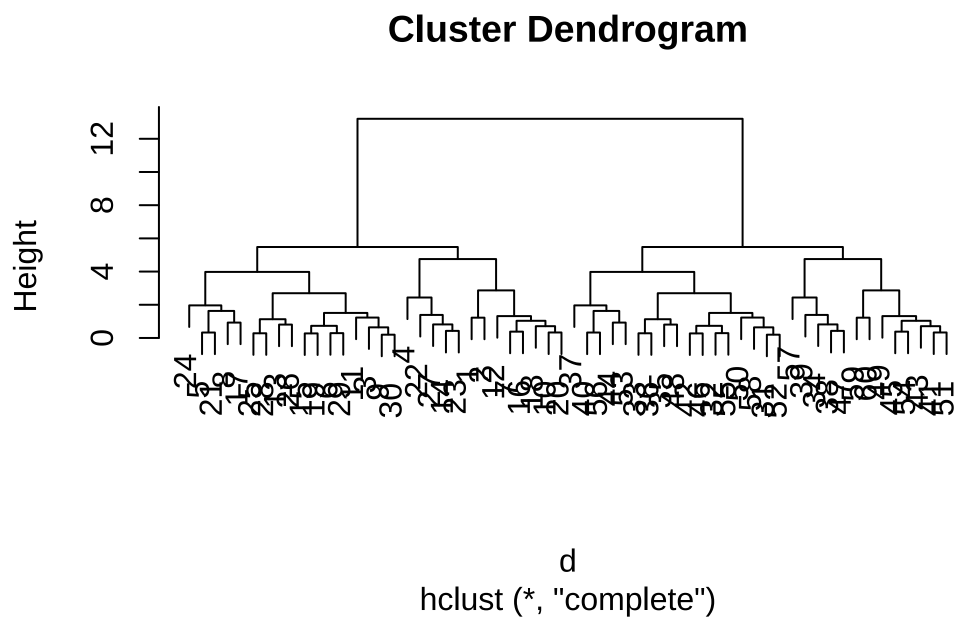
We can us the ‘hclust()’ function for Hieraarchical Clustering. Unlike ‘kmeans()’, where we could just pass in our data as input, we need to give ‘hclust()’ a “distance matrix”.

We will use the ‘dist()’ function to start with.

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

```
Call:
hclust(d = d)

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

```r
plot(hc)
```



I can now “cut” my tree with the ‘cutree()’ to yield a cluster membership vector.

```r
grps <- cutree(hc, h=8)
```

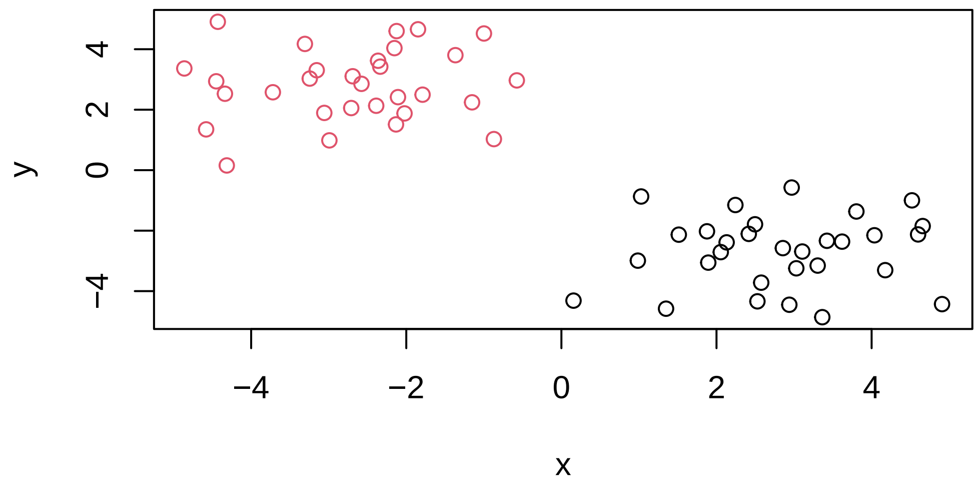
You can also tell “cutree()’ to cut where it yields”k” groups.

```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 2 2
```

Plot of x colored by groups

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



Principal Component Analysis (PCA)

We will import the data provided from the UK food. 'row.names =1" removes the first incorrect column, so the first column is counted as names and not part of the data.

```
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?

```
dim(x)
```

```
[1] 17  5
```

```
head(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
```

The first column in incorrectly used in the data.

```
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
```

Instead we should use 'row.names=1'.

```
 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
```

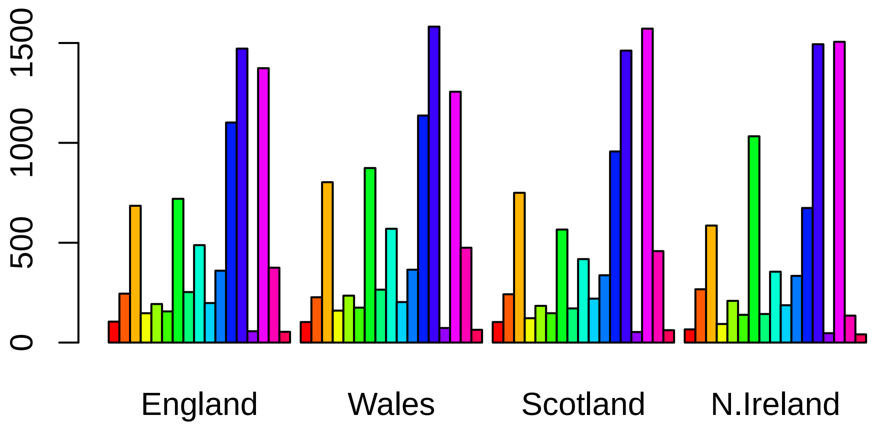
> Q2: Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

The second method is better because the first one may delete the next column if it is run again.

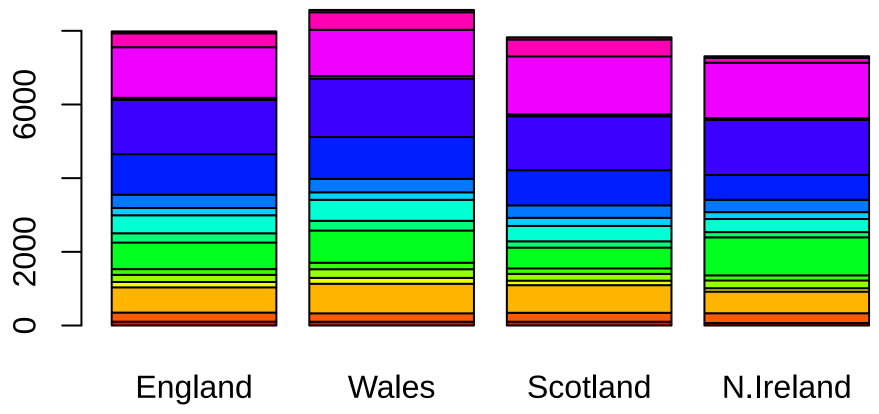
> Q3: Changing what optional argument in the above barplot() function results in the following plot?

Now we will plot

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



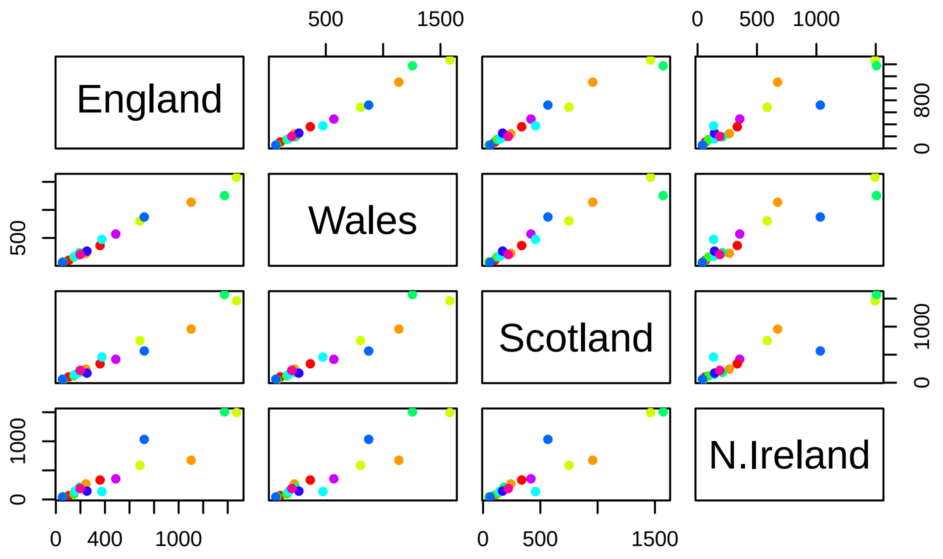
```r
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



Using ‘beside=F’ places the rows of each column on top of each other instead of beside each other.

> 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)
```



This plot creates a graph to compare each country with one another. If the points are the the diagonal, the food consumption is generally the same.

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

The food consumption for N. Ireland compared to other countries is not on the diagonal.

Using PCA

```
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
```

```
attributes(pca)
```

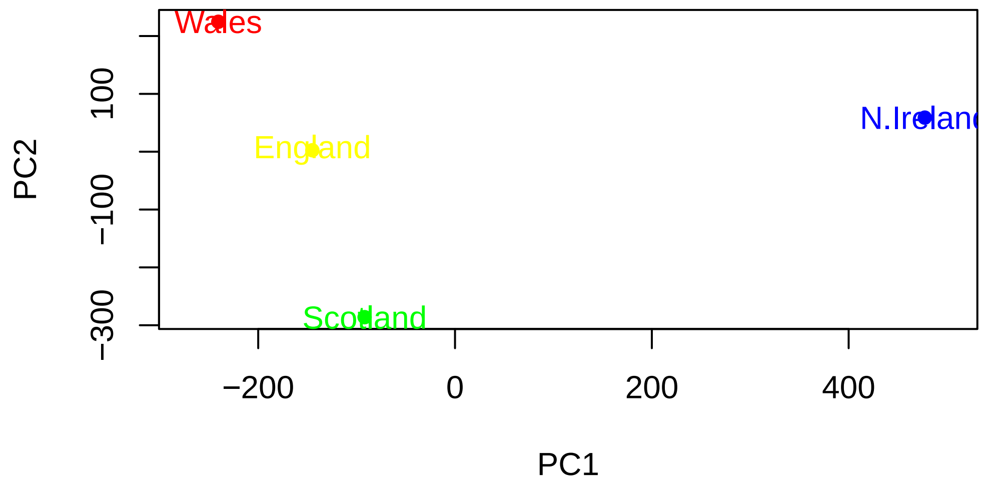
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```
pca$x
```

```
                PC1         PC2         PC3           PC4
England   -144.99315    2.532999 -105.768945  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
```

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

```
paint <- c("yellow", "red", "green", "blue")
plot(pca$x[,1], pca$x[,2],
     col = paint,
     pch=16,
     xlab="PC1", ylab="PC2", xlim=c(-270,500))
text(pca$x[,1], pca$x[,2], colnames(x), col = paint)
```


Wales
England
N.Irelan
Scotland
PC2
100
−100
−300
−200
0
200
400
PC1