Class 7: Clustering and PCA

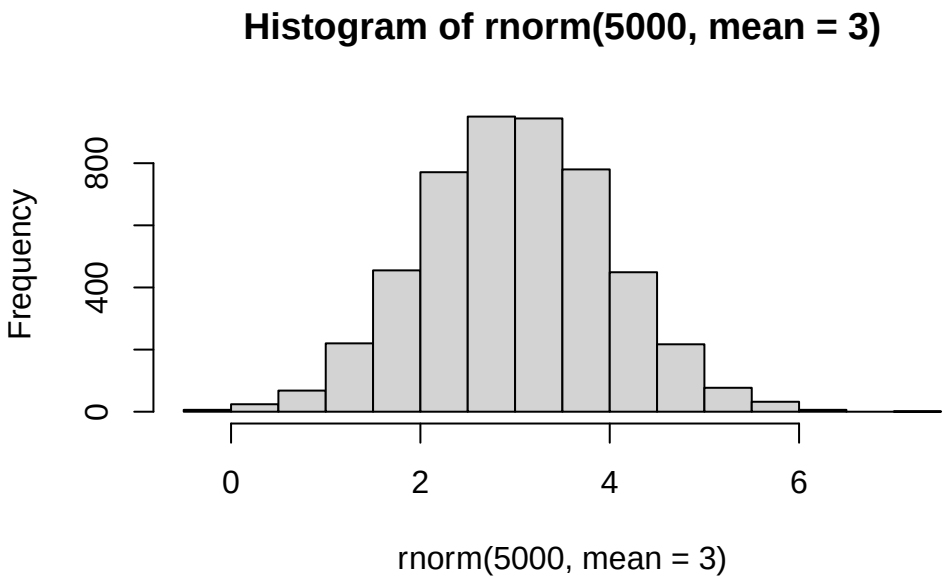
Rogelio Castro

Clustering

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

We can use the `rnorm()` function to fet random numbers from a a normal distribution around a given `mean`. And `hist()` to get a histogram of said data.

```
hist( rnorm(5000, mean = 3) )
```



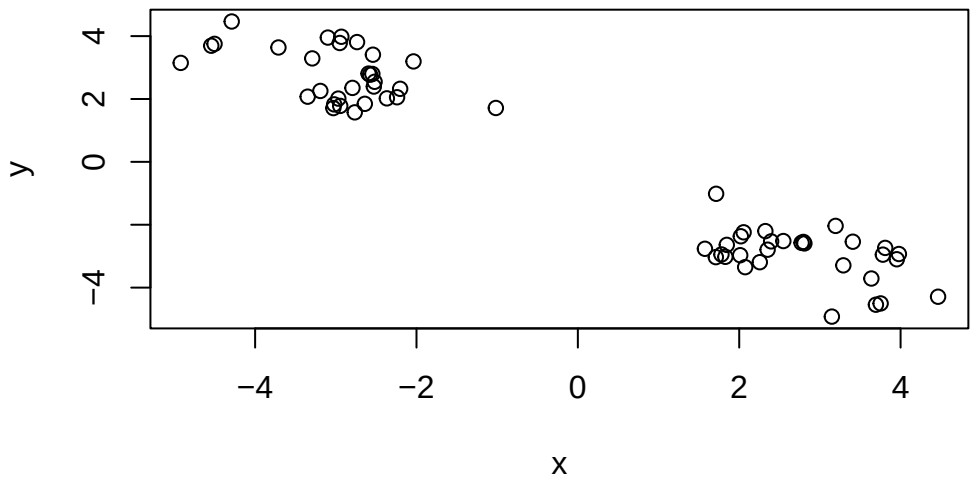
Let's get 30 point with a mean of 3 and another 30 wiht a mean of -3.

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

```
 [1] -2.367145 -3.711015 -3.020383 -4.543616 -4.922213 -2.528742 -2.931254
 [8] -2.037128 -2.240635 -3.191681 -2.640558 -4.503220 -2.951590 -2.736913
[15] -3.098667 -2.202498 -3.291862 -3.349046 -2.969132 -4.290123 -2.574978
[22] -2.546887 -1.016245 -3.030622 -2.594692 -2.793638 -2.519252 -2.765144
[29] -2.945360 -2.540024  3.408222  1.779858  1.575027  2.546725  2.352145
[36]  2.807946  1.710076  1.713970  2.795787  2.772730  4.464823  2.012252
[43]  2.073666  3.290360  2.324021  3.953130  3.809531  3.780664  3.752862
[50]  1.845921  2.255368  2.054764  3.194468  3.979254  2.395822  3.148312
[57]  3.694348  1.828305  3.638279  2.019745
```

Put tow of these together:

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



K-means clustering.

Very popular clusterring method that we can use witht 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  2.765946 -2.961809
2 -2.961809  2.765946

Clustering vector:
 [1] 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 2 1 1 1 1 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

Within cluster sum of squares by cluster:
[1] 39.47945 39.47945
 (between_SS / total_SS =  92.6 %)

Available components:

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

Q1. 30 and 30. Q2. `km$size` for cluster size

```r
km$size
```

```
[1] 30 30
```

`km$cluster` for cluster assingment

```r
km$cluster
```

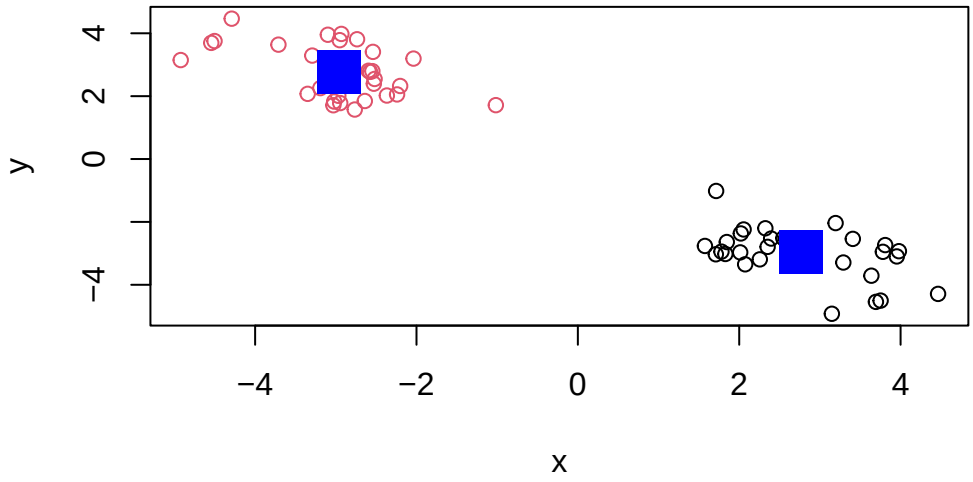
```
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

`km$centers` for centers

```r
km$centers
```

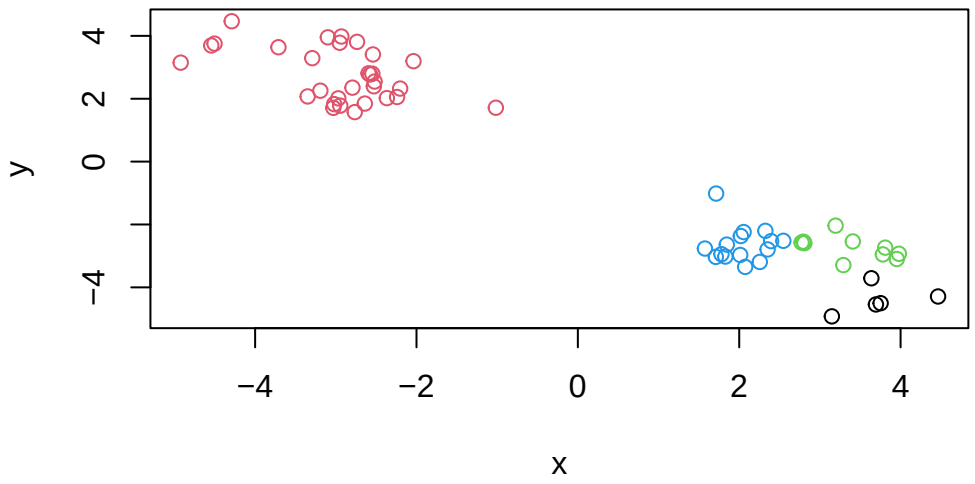
```
          x         y
1  2.765946 -2.961809
2 -2.961809  2.765946
```

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



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

```
km <- kmeans(x, centers=4)
plot(x, col=km$cluster)
```



#Hierarchical Clustering

We can use `hcluster()` function for Hierarchical Clustering. Unlike `kmeans()`, where we could just pass in our data as input, we need to give `hclust()` a "distance matrix".

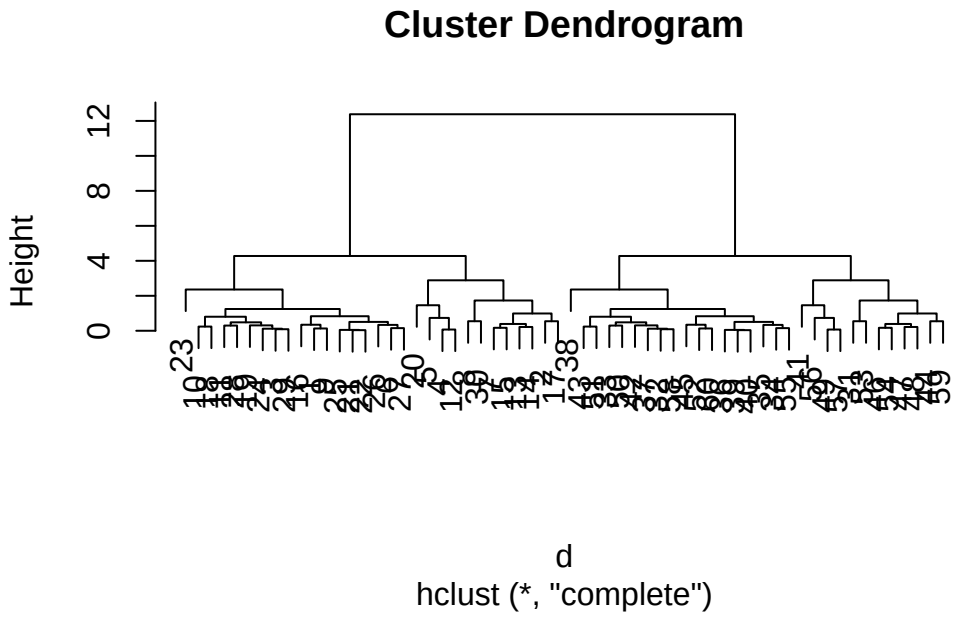
We will use `dist()` function to start with/

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

```
Call:
hclust(d = d)

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

```
plot(hc)
```



I can now "cut" my tree with the `cutree()` to yield a cluster membership vector.

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

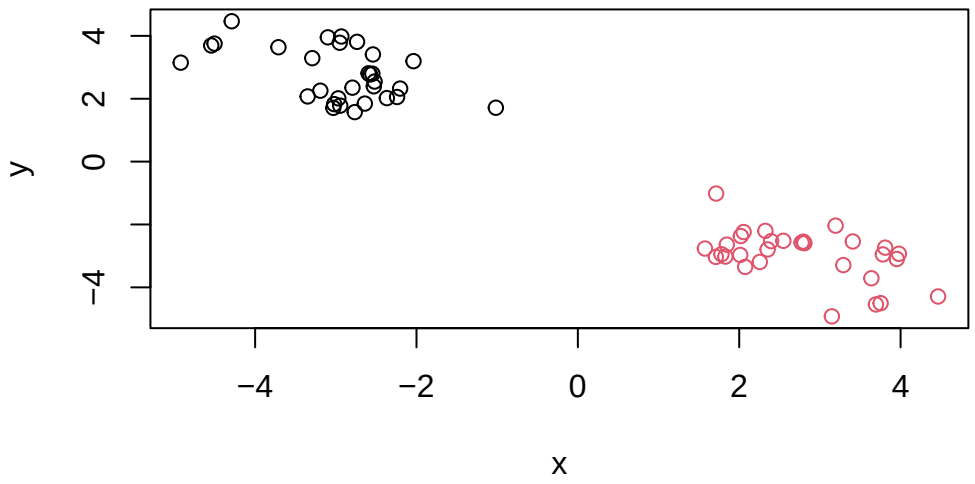
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

You can also tell `cutree()` to cut where it yields "k" groups.

```
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 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
```

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



Principal Component Analysis (PCA)

Use `row.names` to set the names of the foods to the columns

```
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? 17 rows and 5 columns but we want 4 columns, so we use `row.names` to eliminate the extra column by setting the food name to the row name.

```
dim(x)
```

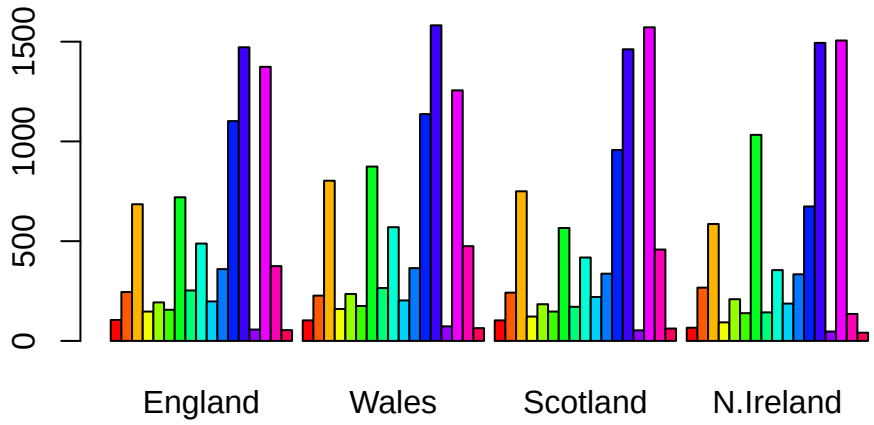
```
[1] 17  4
```

```
View(x)
```

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

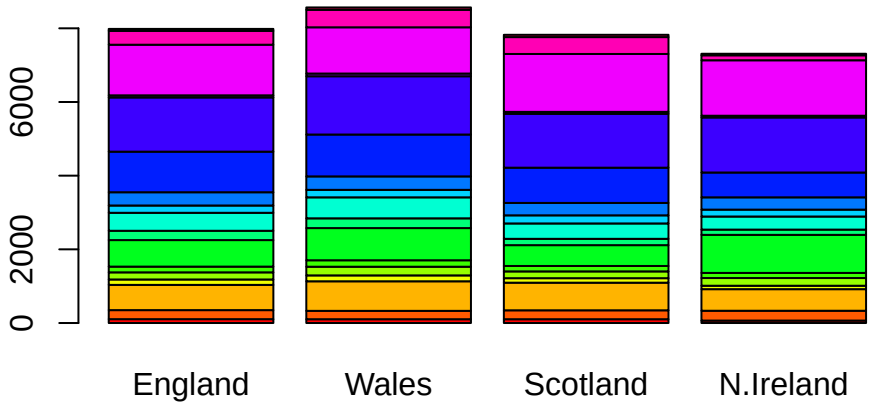
circumstances? I like the renaming of the rows, because it is simpler and can eliminate unnecessary rows.

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



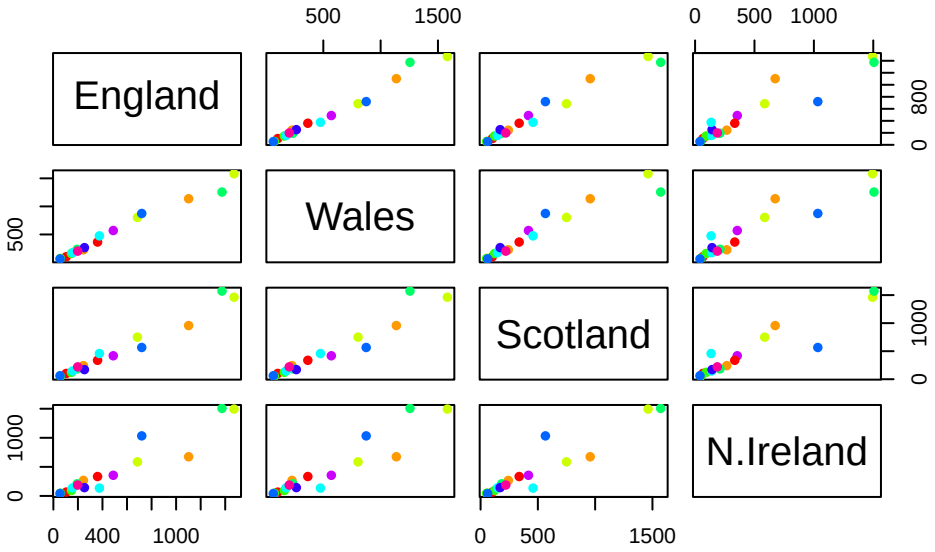
Q3: Changing what optional argument in the above barplot() function results in the following plot? Change 'beside=T' to 'beside=F'

```
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)
```



Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set? The blue dot is considerably lower than the other countries.

#PCA to the rescue!

The main PCA funtion in base R is called `prcomp()` it expects the transpose of our data.

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

```
Importance of components:
                            PC1      PC2      PC3        PC4
Standard deviation     324.1502 212.7478 73.87622 5.552e-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.

```
attributes(pca)
```

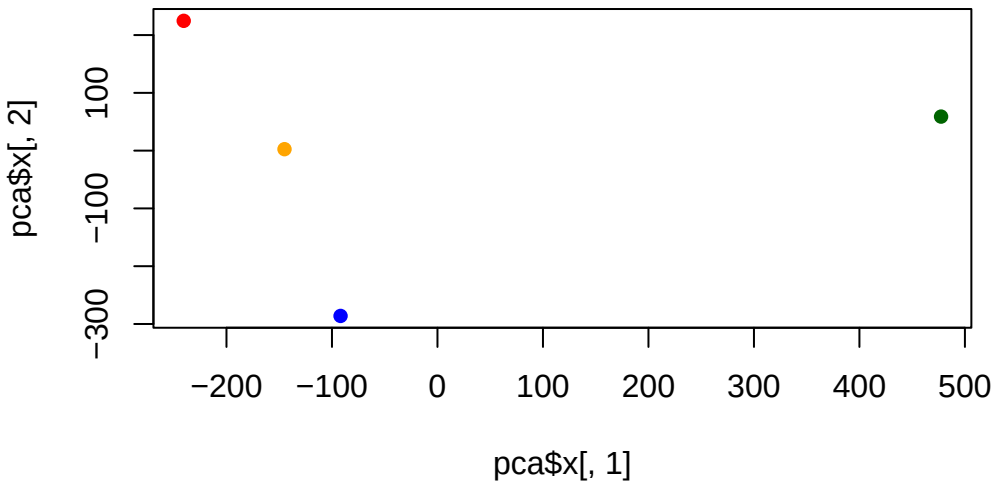
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

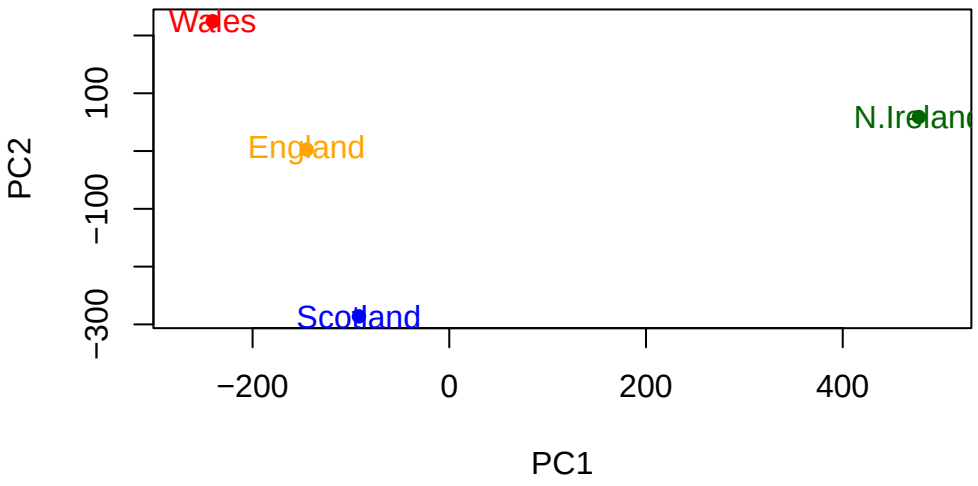
```
pca$x
```

```
                PC1         PC2         PC3           PC4
England   -144.99315    2.532999 -105.768945  1.042460e-14
Wales     -240.52915  224.646925   56.475555  9.556806e-13
Scotland   -91.86934 -286.081786   44.415495 -1.257152e-12
N.Ireland  477.39164   58.901862    4.877895  2.872787e-13
```

```
plot(pca$x[,1], pca$x[,2],
     col=c("orange", "red", "blue", "darkgreen"),
     pch=16)
```




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


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