Class 7: Clustering and PCA

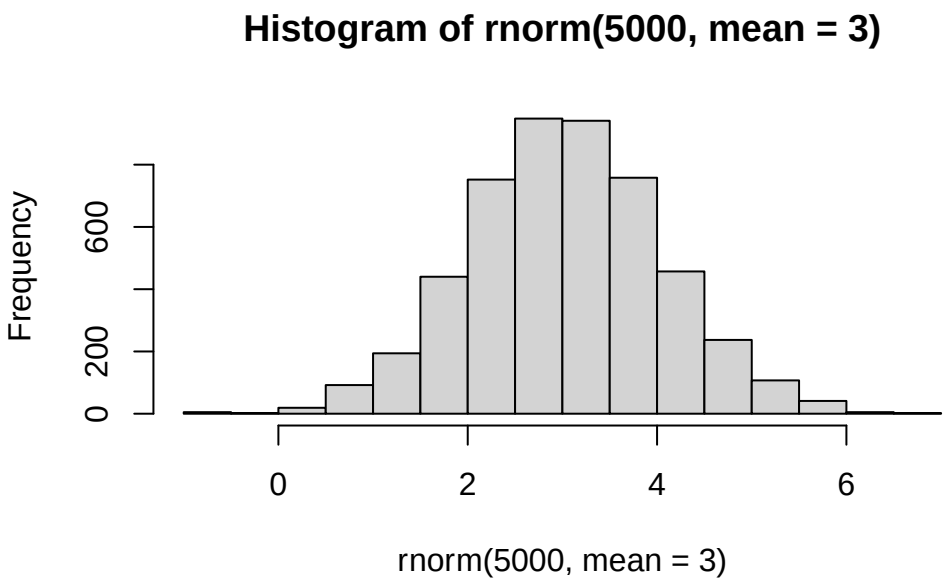
Emily Rodriguez

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 thr `rnorm()` function to get random numbers from a normal distribution around a given `mean`.

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



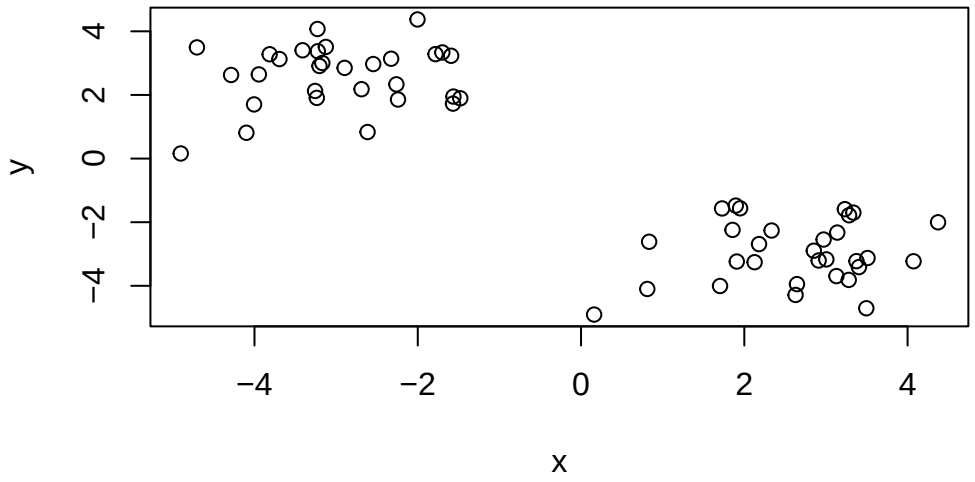
Let's get 30 points with a mean of 3.

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

```
 [1]  3.4043081  2.6453314  4.3736647  2.1260965  2.6280147  3.5096791
 [7]  0.8115729  0.1602921  3.0050490  2.9715440  3.3738790  0.8340545
[13]  1.7029188  2.1805208  1.8954765  1.9080243  1.8558031  2.8512581
[19]  2.3340832  2.9109475  3.3361616  1.9485786  3.2791660  4.0730535
[25]  3.1285262  1.7276455  3.4945588  3.1387791  3.2319511  3.2833916
[31] -1.7823997 -1.5914150 -2.3263770 -4.7052249 -1.5681594 -3.6944528
[37] -3.2283482 -3.8142158 -1.5632480 -1.6991367 -3.2045431 -2.2618866
[43] -2.8960956 -2.2420561 -3.2366538 -1.4791199 -2.6888152 -4.0053792
[49] -2.6149348 -3.2244908 -2.5454872 -3.1694701 -4.9037080 -4.0993014
[55] -3.1262668 -4.2869589 -3.2580749 -2.0031471 -3.9465489 -3.4108159
```

Put two of these together:

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



K-means clustering.

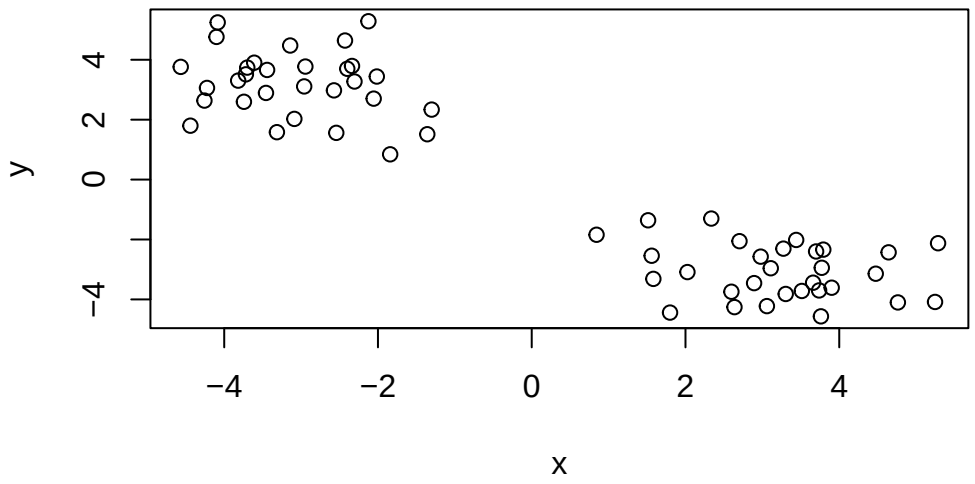
Very popular clustering method that we can use with the `kmeans()` function in base R.

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

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

Cluster means:
          x         y
1 -2.952558  2.604144
2  2.604144 -2.952558

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

Within cluster sum of squares by cluster:
[1] 54.82952 54.82952
 (between_SS / total_SS =  89.4 %)

Available components:

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

Generate some example data for clustering

```
tmp <- c(rnorm(30, -3), rnorm(30, 3))
x <- data.frame(x = tmp, y = rev(tmp))
plot(x)
```



Q. How many points are in each cluster?

Q. What `component` of your result object details?

- cluster size?

```
km$size
```

```
[1] 30 30
```

- cluster?

```
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 1 1
```

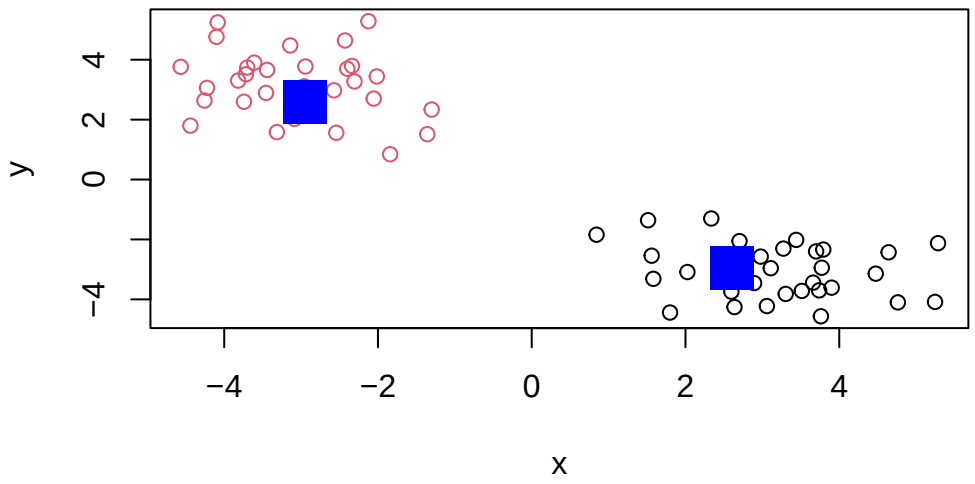
- cluster center?

```
km$centers
```

```
          x         y
1 -2.952558  2.604144
2  2.604144 -2.952558
```

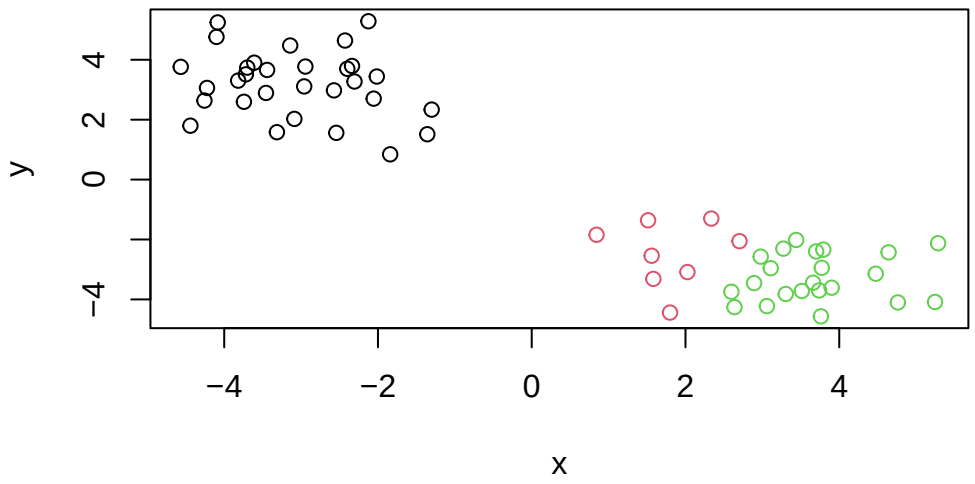
Q. Plot x colored by the kmeans cluster assignment and add cluster centers as blue points.

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



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

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



Hierarchical Cluster

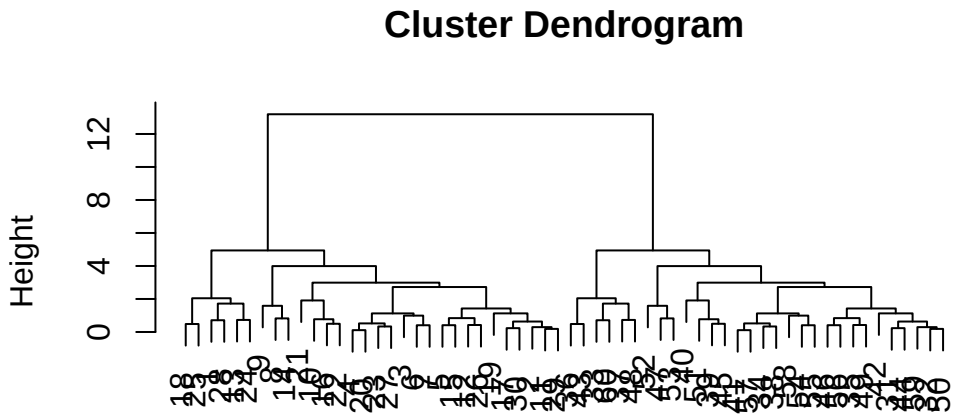
We can use the `hclust()` function for Hierarchical Clustering. Unlike `kmeans()`, where we could just pass in our data as input, we need to give `hclust()` a "distance matrix".

We will use 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
```

```
plot(hc)
```



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

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

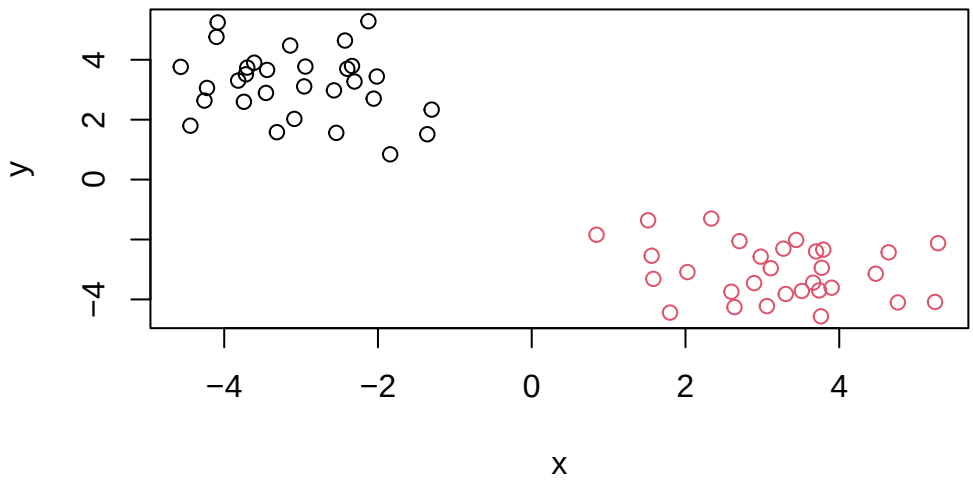
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

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)
```



Principle Component Analysis (PCA)

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

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

```
dim(x)
```

```
[1] 17  4
```

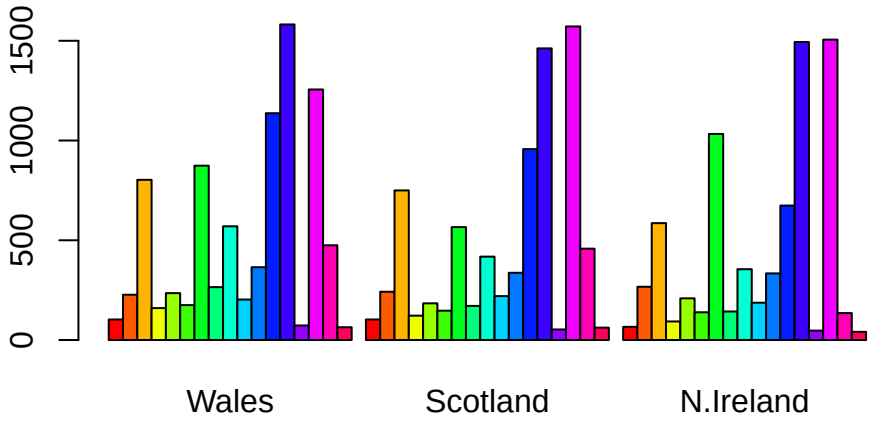
```
x <- x[,-1]
head(x)
```

```
               Wales Scotland N.Ireland
Cheese           103      103        66
Carcass_meat     227      242       267
Other_meat       803      750       586
Fish             160      122        93
Fats_and_oils    235      184       209
Sugars           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?

```
I prefer using the `read.csv()` function because the `rownames()` function removes      Engla
```

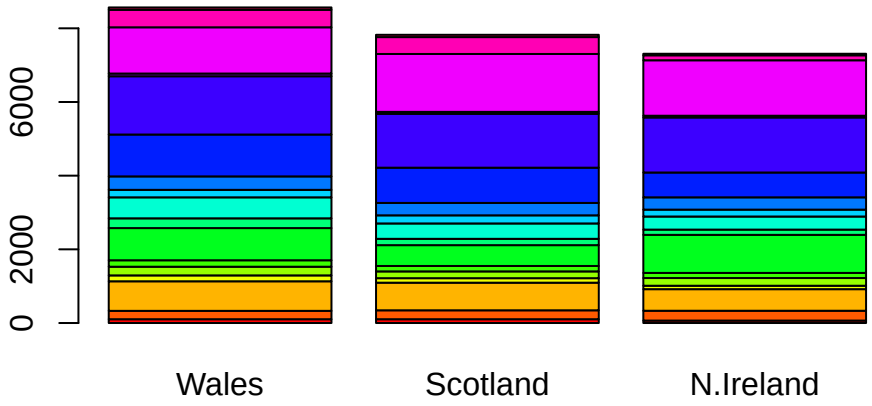
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

Changing the argument `beside` to `F` will result in the following bar plot.

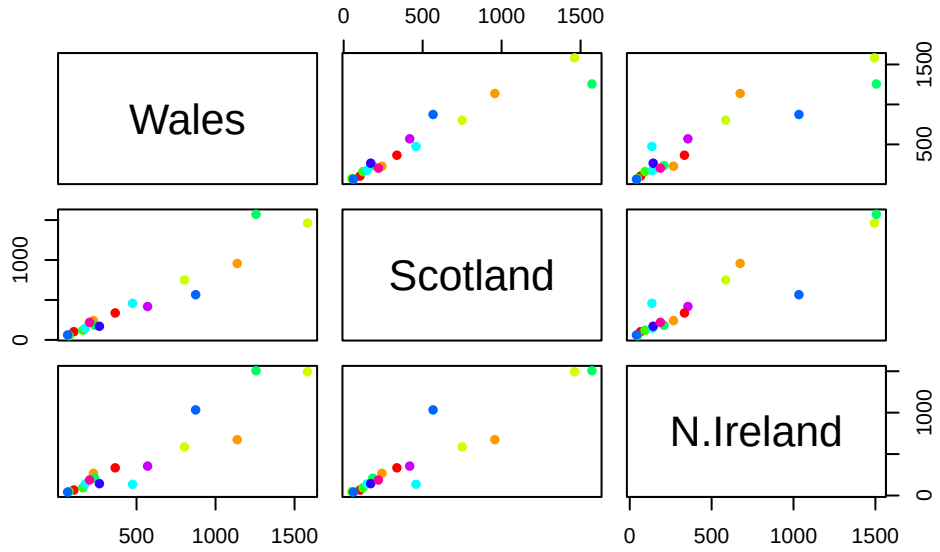
```
barplot(as.matrix(x), beside= F, col=rainbow(nrow(x)))
```



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?

```
Some countries lie on the y-axis and some on the x-axis. It means that both          count
```

```
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 main difference is the blue point.
```

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

```
The main PCA function 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
Standard deviation     379.8991 260.5533 1.515e-13
Proportion of Variance   0.6801   0.3199 0.000e+00
Cumulative Proportion    0.6801   1.0000 1.000e+00
```

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

```
pca$x
```

```
                PC1        PC2            PC3
Wales     -288.9534  226.36855  2.296774e-14
Scotland  -141.3603 -284.81172  4.517428e-13
N.Ireland  430.3137   58.44317 -1.407069e-13
```

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