Class 7: Clustering and PCA

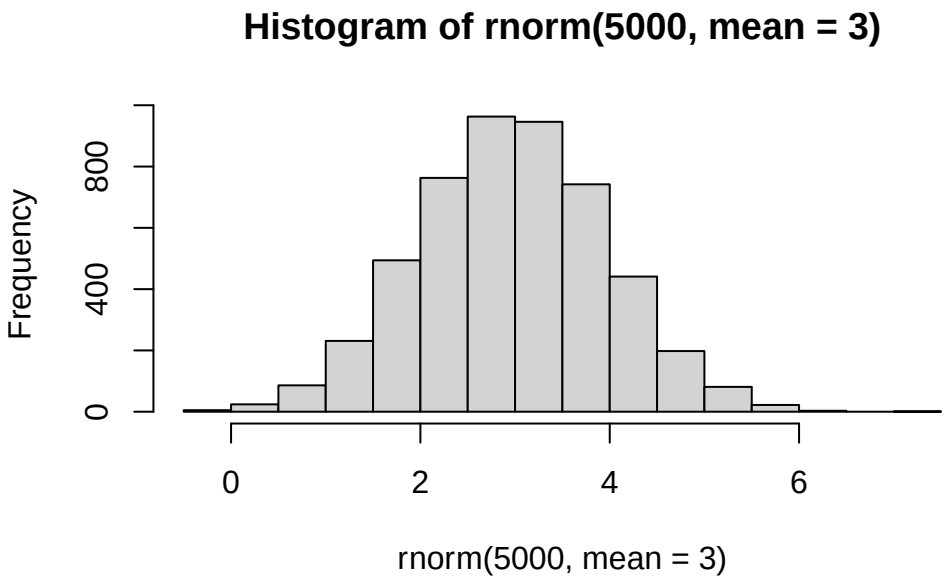
Kaitlyn Madriaga

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' with a given 'standard distribution'.

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



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

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

```
 [1]  6.0431902  4.5829073  0.8692819  2.7942317  1.3978066  1.8910425
 [7]  6.2054051  2.3088192  3.5060493  2.8924746  1.5941129  3.6109252
[13]  2.4343152  3.4886846  4.3706531  3.4203120  3.2253221  3.5745921
[19]  3.3433835  2.9317302  2.2021664  2.0552990  2.0159434  2.4909327
[25]  3.8557910  2.3746783  2.9251639  2.2084081  4.4906699  2.9306999
[31] -1.3573441 -3.0763620 -3.0595074 -2.4838478 -2.8975733 -5.4338971
[37] -2.2350316 -3.2473953 -3.6615768 -3.5146266 -3.7003850 -1.0201808
[43] -2.4527687 -4.7112421 -3.5126259 -2.1072661 -2.4575815 -3.1452691
[49] -2.5372771 -3.0108466 -3.7828865 -3.1320372 -1.2897677 -3.9774091
[55] -3.1490381 -3.3802300 -2.6778517 -1.9969758 -2.0597000 -1.7341066
```

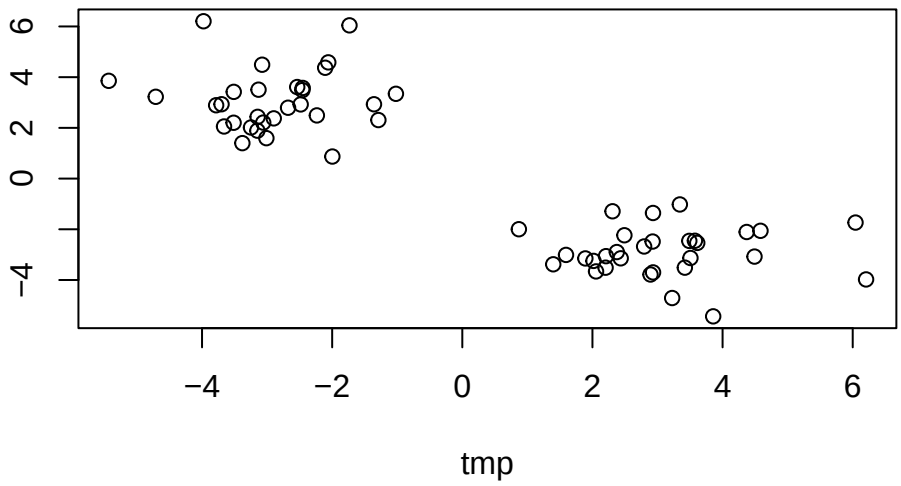
Put the two of these together:

```r
x <- cbind(tmp, rev(tmp))
x
```

```
            tmp
 [1,]  6.0431902 -1.7341066
 [2,]  4.5829073 -2.0597000
 [3,]  0.8692819 -1.9969758
 [4,]  2.7942317 -2.6778517
 [5,]  1.3978066 -3.3802300
 [6,]  1.8910425 -3.1490381
 [7,]  6.2054051 -3.9774091
 [8,]  2.3088192 -1.2897677
 [9,]  3.5060493 -3.1320372
[10,]  2.8924746 -3.7828865
[11,]  1.5941129 -3.0108466
[12,]  3.6109252 -2.5372771
[13,]  2.4343152 -3.1452691
[14,]  3.4886846 -2.4575815
[15,]  4.3706531 -2.1072661
[16,]  3.4203120 -3.5126259
[17,]  3.2253221 -4.7112421
[18,]  3.5745921 -2.4527687
[19,]  3.3433835 -1.0201808
[20,]  2.9317302 -3.7003850
[21,]  2.2021664 -3.5146266
```

```
[22,]  2.0552990 -3.6615768
[23,]  2.0159434 -3.2473953
[24,]  2.4909327 -2.2350316
[25,]  3.8557910 -5.4338971
[26,]  2.3746783 -2.8975733
[27,]  2.9251639 -2.4838478
[28,]  2.2084081 -3.0595074
[29,]  4.4906699 -3.0763620
[30,]  2.9306999 -1.3573441
[31,] -1.3573441  2.9306999
[32,] -3.0763620  4.4906699
[33,] -3.0595074  2.2084081
[34,] -2.4838478  2.9251639
[35,] -2.8975733  2.3746783
[36,] -5.4338971  3.8557910
[37,] -2.2350316  2.4909327
[38,] -3.2473953  2.0159434
[39,] -3.6615768  2.0552990
[40,] -3.5146266  2.2021664
[41,] -3.7003850  2.9317302
[42,] -1.0201808  3.3433835
[43,] -2.4527687  3.5745921
[44,] -4.7112421  3.2253221
[45,] -3.5126259  3.4203120
[46,] -2.1072661  4.3706531
[47,] -2.4575815  3.4886846
[48,] -3.1452691  2.4343152
[49,] -2.5372771  3.6109252
[50,] -3.0108466  1.5941129
[51,] -3.7828865  2.8924746
[52,] -3.1320372  3.5060493
[53,] -1.2897677  2.3088192
[54,] -3.9774091  6.2054051
[55,] -3.1490381  1.8910425
[56,] -3.3802300  1.3978066
[57,] -2.6778517  2.7942317
[58,] -1.9969758  0.8692819
[59,] -2.0597000  4.5829073
[60,] -1.7341066  6.0431902
```

```
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:
        tmp
1  3.067833 -2.893420
2 -2.893420  3.067833

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

Within cluster sum of squares by cluster:
[1] 70.65189 70.65189
 (between_SS / total_SS =  88.3 %)
```

```
Available components:

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

```
km$size
```

```
[1] 30 30
```

Q. How many points are in each cluster?

30

Q. What 'component' of your result object details

- cluster size?

```
km$size
```

```
[1] 30 30
```

- cluster assignment/membership?

```
km$cluster
```

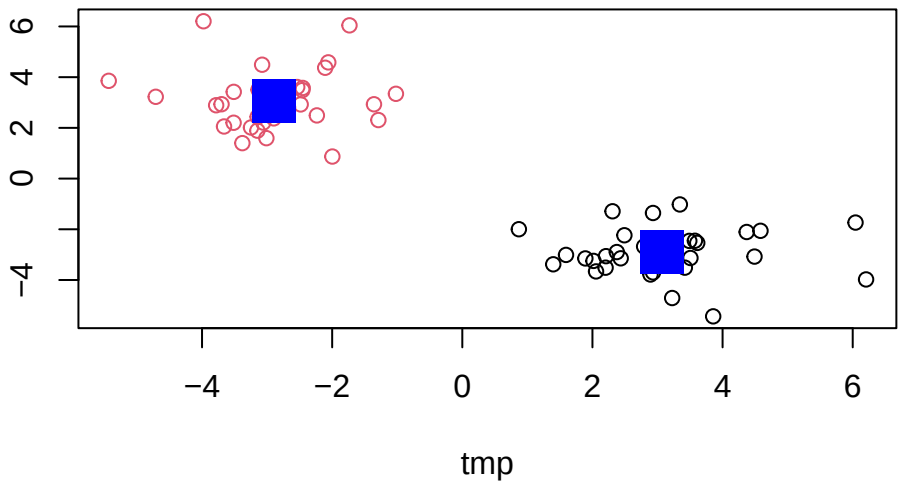
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

- cluster center?

```
km$centers
```

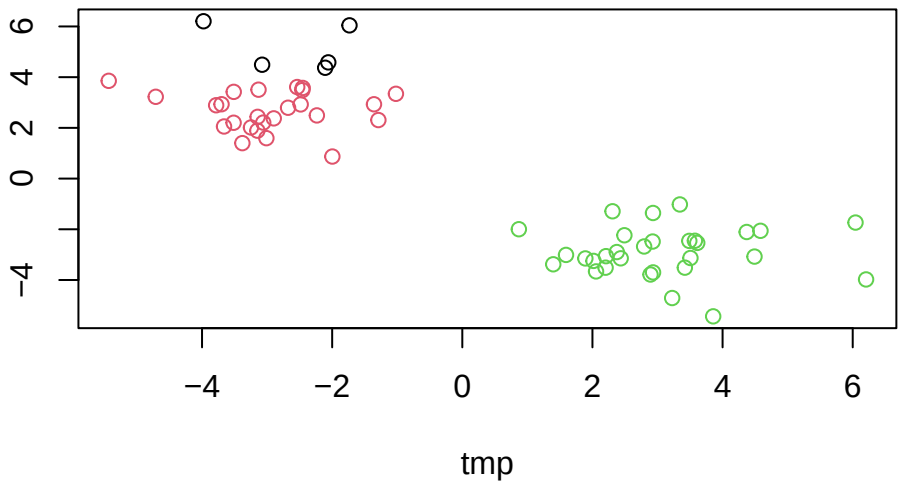
```
        tmp
1  3.067833 -2.893420
2 -2.893420  3.067833
```

```
mycols <- km$cluster
plot(x, col = mycols)
points(km$center, 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 Clustering

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".

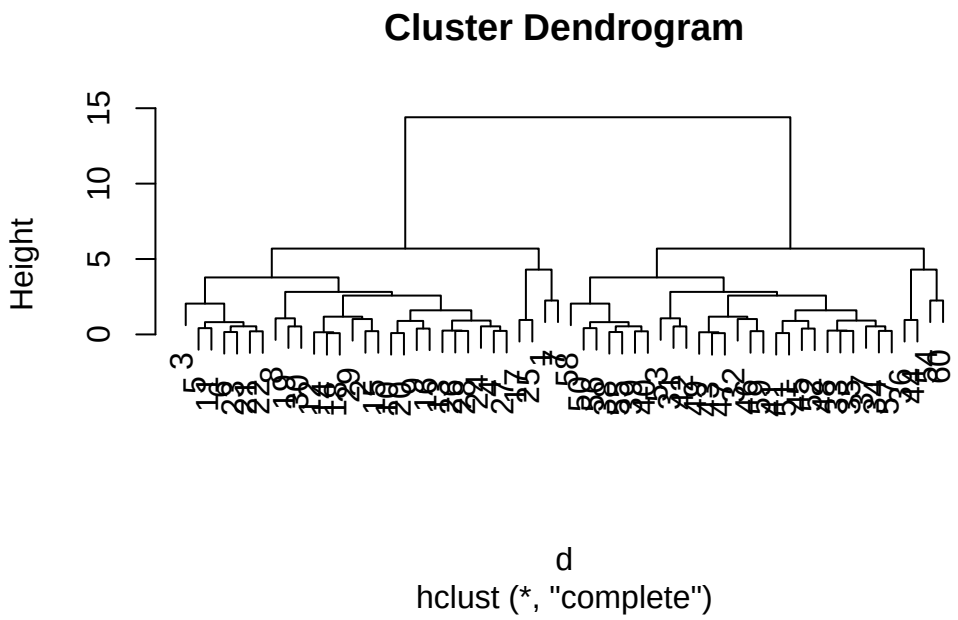
We will use the 'dist()' function to start with.

```
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
```

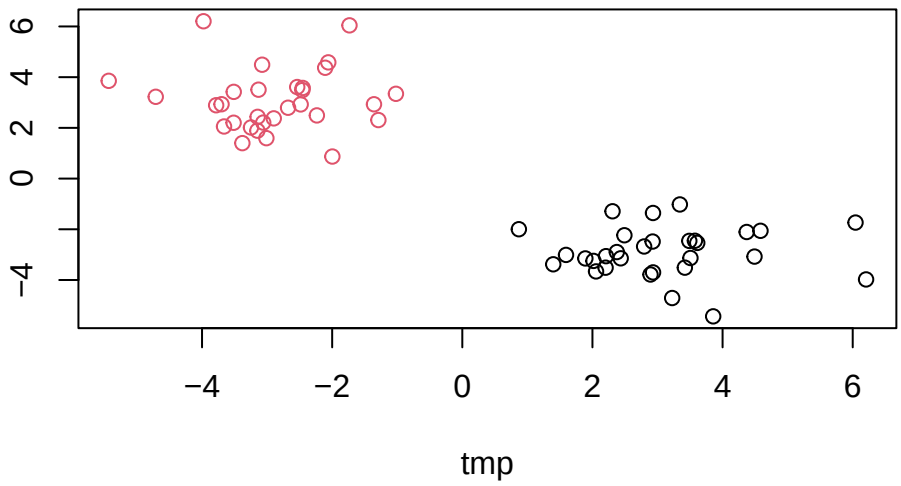
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

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)
```



Principal Component Analysis (PCA)

##PCA of UK food data

```
url <- "https://tinyurl.com/UK-foods"
#use row.names to remove the first column, so that the food names aren't a column
x <- read.csv(url, row.names=1)
dim(x)
```

```
[1] 17  4
```

```
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
```

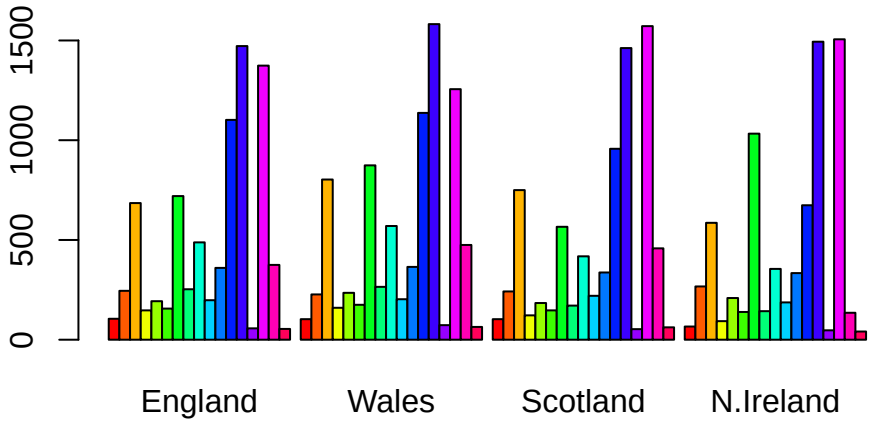
Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

17 rows and 4 columns

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?

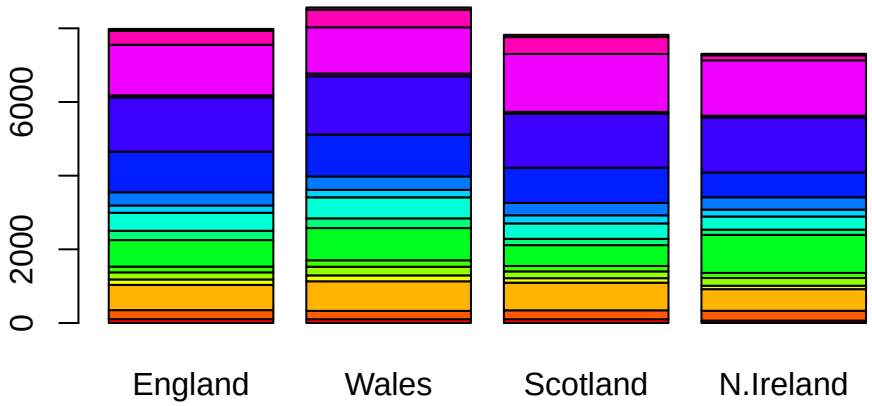
Using row.names in read.csv is better because the info can be altered when using the other approach

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



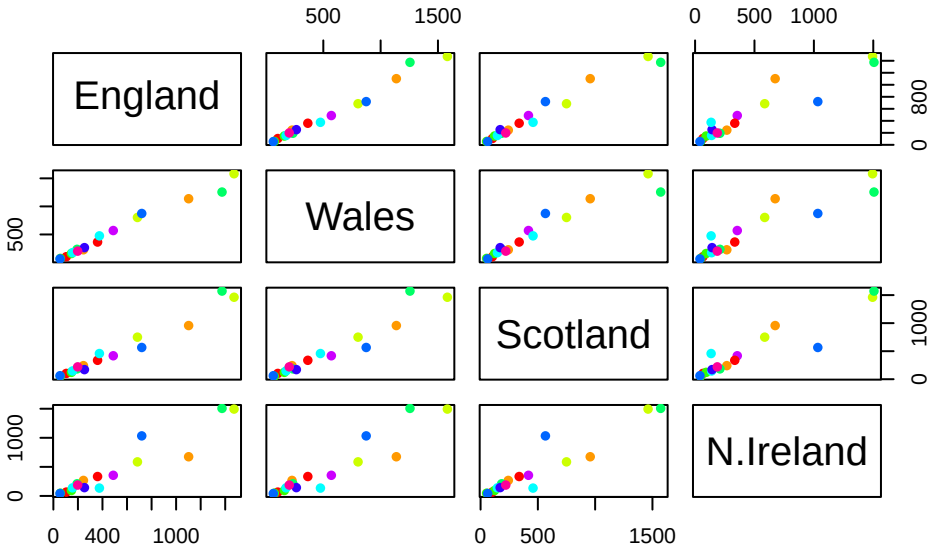
Q3: Changing what optional argument in the above barplot() function results in the following plot?

```
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)
```



These are pairs of countries plotted against each other. The country going across is the y-axis for the plot, while the country going down is the x-axis for the plot. e.g. the plot with "England" to the left and "Wales" below is a plot of Wales x England, where Wales = x-axis and England = y-axis. The plot where "Wales" is to the right of the plot has the reverse, where Wales = y-axis and England = x-axis.

If the point lies on the diagonal, it means both countries have the same amount. If it is above the diagonal, the country on the y-axis has more of it than the country on the x-axis.

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

The blue point

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       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"
```

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

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