Class 7: Machine Learning 1

Bernice Lozada (A16297973)

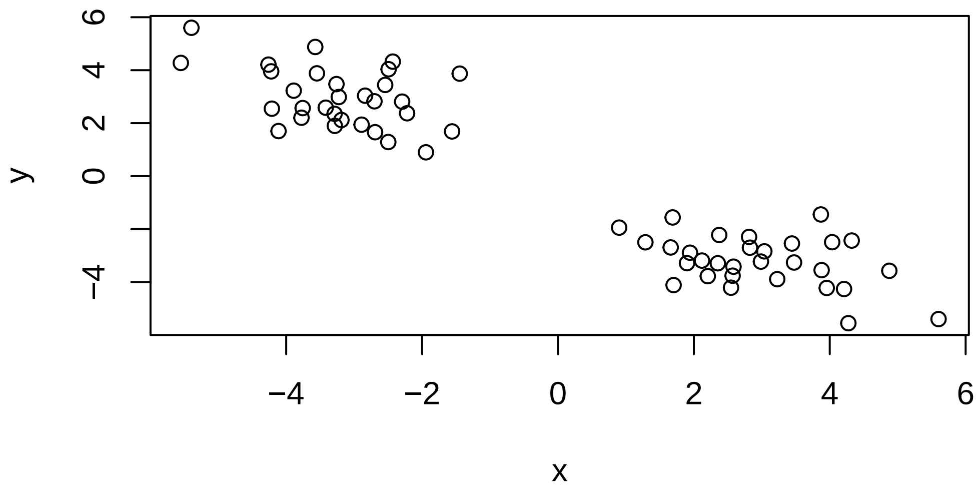
Today we will start our multi-part exploratoin of some key machine learning methods. We will begin with clustering - finding groupings in data, and then dimensionality reduction.

Clustering

Let's start with "k-means" clustering.

The main function in base R for this is `kmeans()`.

```r
# Make up some data
# Make two separated clusters of data
tmp <- c(rnorm(30,-3), rnorm(30,+3))
x <- cbind(x = tmp, y = rev(tmp))
plot(x)
```



Trying out `kmeans()`

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

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

Cluster means:
          x         y
1  2.955493 -3.217051
2 -3.217051  2.955493

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 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] 64.47866 64.47866
 (between_SS / total_SS =  89.9 %)

Available components:
```

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

Q. How many points in each cluster?

```
km$size
```

```
[1] 30 30
```

Q. What component of your result object details cluster assignment/membership?

```
#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 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
```

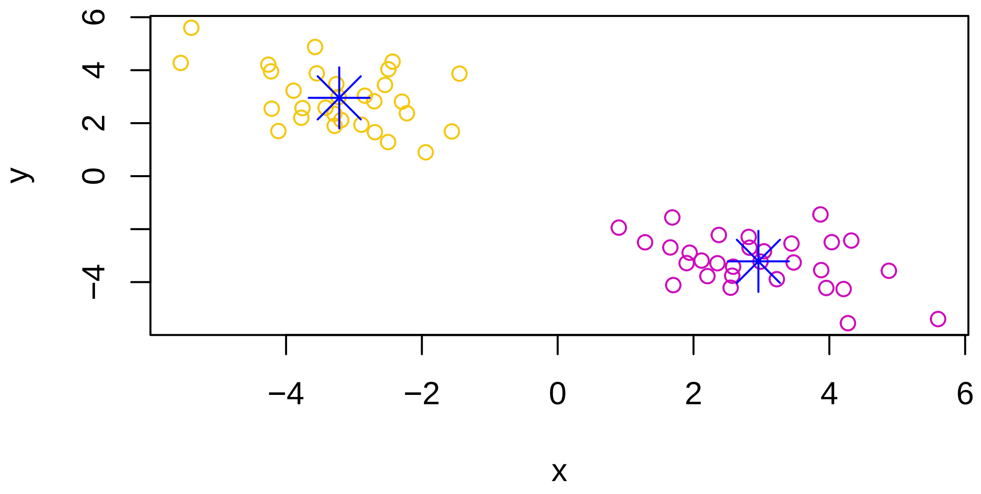
Q. What are centers/mean values of each cluster?

```
km$centers
```

```
          x         y
1  2.955493 -3.217051
2 -3.217051  2.955493
```

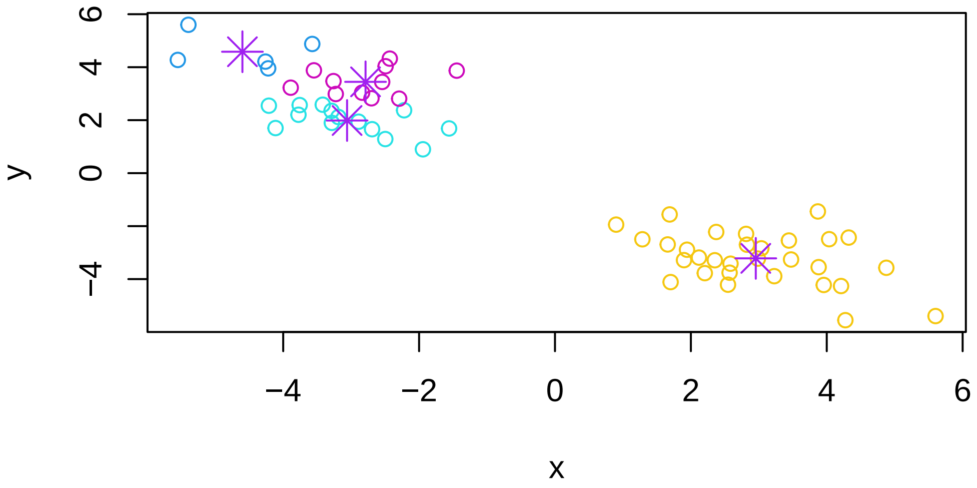
Q. Make a plot of your data showing your clustering results (groupings/clusters and cluster centers)

```
# color by cluster membership
plot(x, col = km$cluster+5)
points(km$centers, col="blue", pch = 8, cex = 3)
```



Q. Run `kmeans()` again and cluster in 4 groups and plot the results.

```r
km <- kmeans(x, centers = 4)
# color by cluster membership
plot(x, col = km$cluster+3)
points(km$centers, col="purple", pch = 8, cex = 2)
```



Biased because the results depend on the number of clusters you specify.

Hierarchal Clustering

This form of clustering aims to reveal the structure in your data by progressively grouping points into an ever smaller number of clusters. Start with every point with its own clusters –> smaller number.

The main function in base R for this is called `hclust()`. This function does not take our input data directly but wants a “distance matrix” that details how (dis)similar all our inputs are to each other.

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

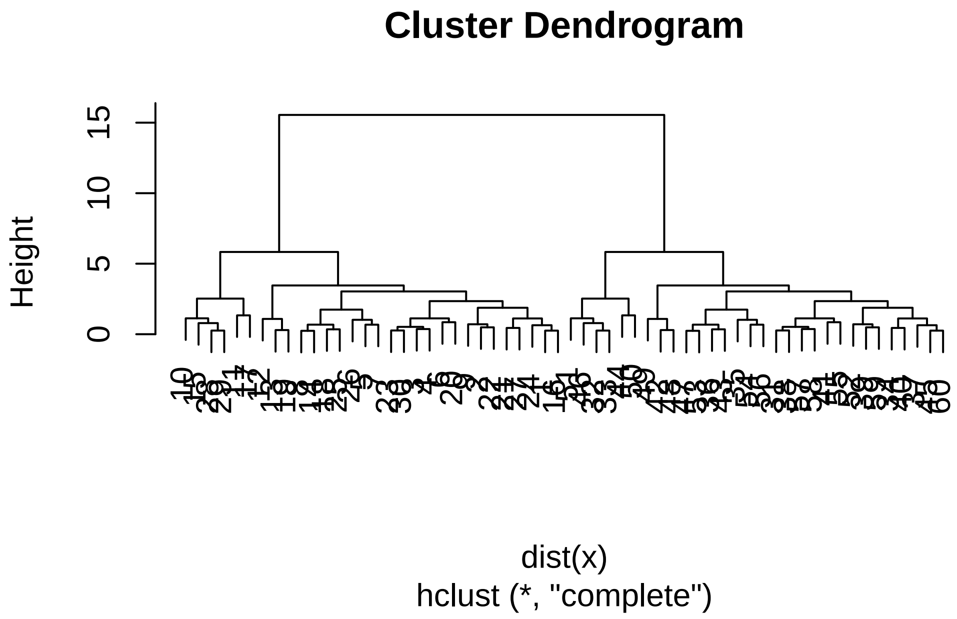
```

Call:
hclust(d = dist(x))

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

The print out above is not very useful, but there is a useful `plot()` method.

```
plot(hc)
```



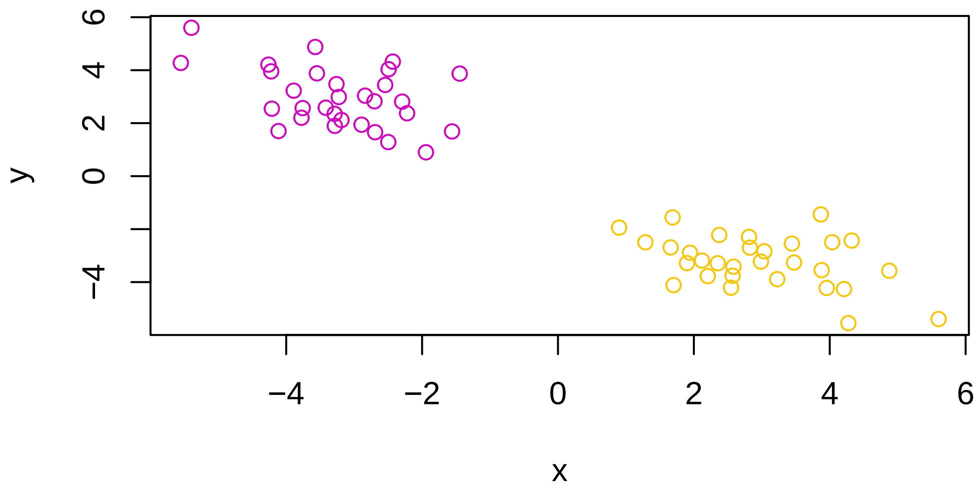
Height of crossbar (horizontal line) indicates how far the points on the goalposts are.

To get my main result (my cluster membership vector), I need to "cut" my tree using the function `cutree()`

```
grps <- cutree(hc, h = 6)
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 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(x, col = grps+5)
```



Principal Component Analysis

The goal of PCA is to reduce the dimensionality of a dataset down to some smaller subset of new variables (called PCs) that are useful bases for further analysis, like visualization, clustering, etc.

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

View(x)
```

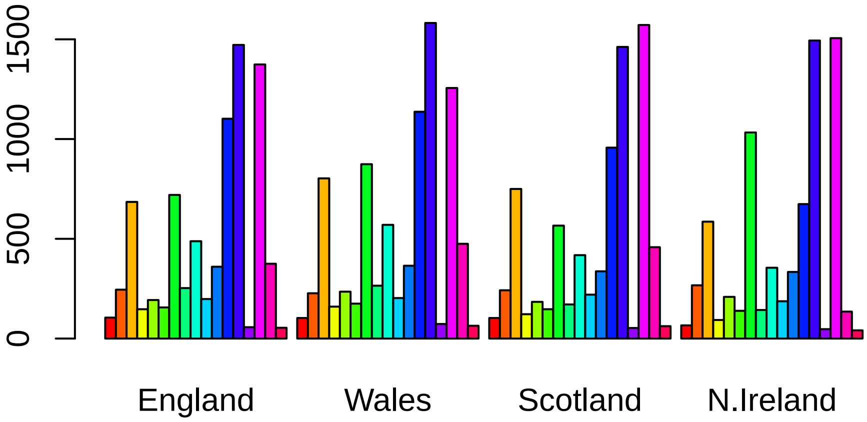
> Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

I could use `nrow()` and `ncol()` to determine the number of rows and columns, respectively.

> 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 would prefer setting row.names = 1, as it requires only running line of code to change the name of the entire data set rather than rerunning the code.

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

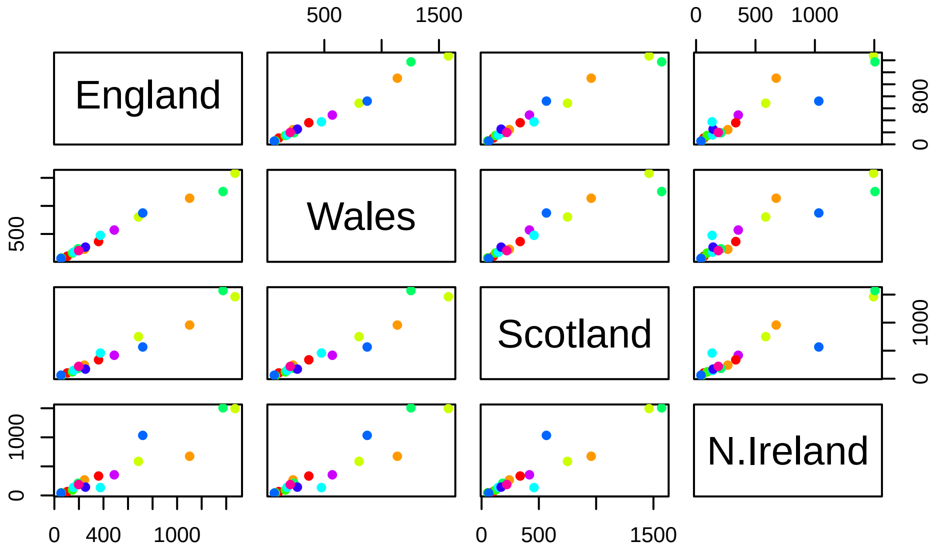


> Q3: Changing what optional argument in the above barplot() function results in the following plot?

Remove the beside = T argument/setting it equal to F would result in the plot on the website.

The so-called “pairs” plot can be useful for small datasets:

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



> 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?

The code leads to a pairwise graphing on different countries against each other. The position of the words indicate which country is on which axes for a given plot (e.g. in the same row means that country is on y-axis, in the same column means it's on the x-axis). A point on the diagonal for a given plot means that the value for that point between the two countries in the plot are very similar.

Pairs plot is useful for small datasets, but can be lots of work to interpret and gets intractable for larger datasets –> USE PCA

The main function to do PCA in base R is `prcomp()`

```
# t(x) so the variables are cols and observations are rows

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

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 3.176e-14
```

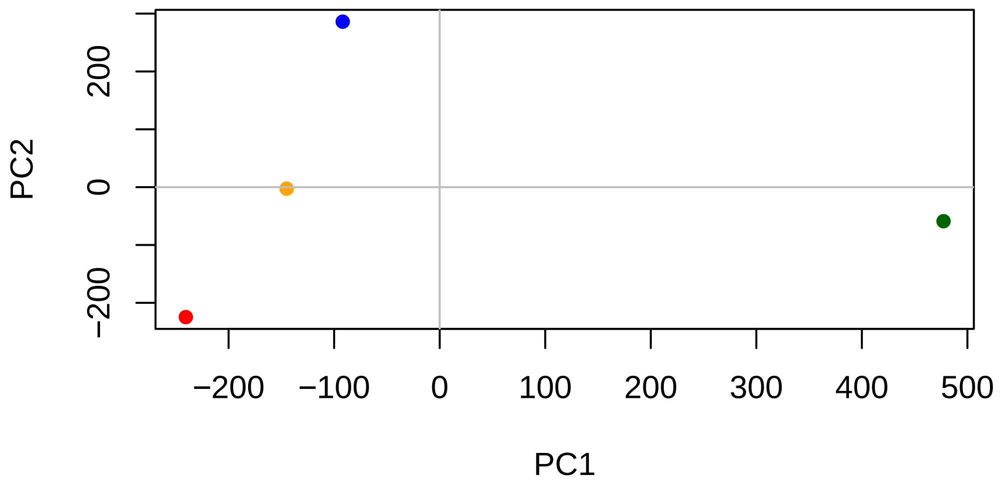
```
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

```
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
```

A major PCA result visualization is called a “PCA plot” (aka a score plot, biplot, PC1 vs PC2 plot, ordination plot)

```
mycols <- c("orange","red","blue","darkgreen")
plot(pca$x[,1],pca$x[,2], col = mycols, pch = 16, xlab = "PC1",
     ylab = "PC2")
abline(h=0, col="gray")
abline(v=0, col="gray")
```



Another output from PCA is called the “loadings” vector or that “rotation” component - this tells us how much the original variables (the foods in this case) contribute to the new PCs.

```
pca$rotation
```

```
                            PC1          PC2         PC3          PC4
Cheese             -0.056955380  0.016012850  0.02394295 -0.694538519
Carcass_meat        0.047927628  0.013915823  0.06367111  0.489884628
Other_meat         -0.258916658 -0.015331138 -0.55384854  0.279023718
Fish               -0.084414983 -0.050754947  0.03906481 -0.008483145
Fats_and_oils      -0.005193623 -0.095388656 -0.12522257  0.076097502
Sugars             -0.037620983 -0.043021699 -0.03605745  0.034101334
Fresh_potatoes      0.401402060 -0.715017078 -0.20668248 -0.090972715
Fresh_Veg          -0.151849942 -0.144900268  0.21382237 -0.039901917
Other_Veg          -0.243593729 -0.225450923 -0.05332841  0.016719075
Processed_potatoes -0.026886233  0.042850761 -0.07364902  0.030125166
Processed_Veg      -0.036488269 -0.045451802  0.05289191 -0.013969507
Fresh_fruit        -0.632640898 -0.177740743  0.40012865  0.184072217
Cereals            -0.047702858 -0.212599678 -0.35884921  0.191926714
Beverages          -0.026187756 -0.030560542 -0.04135860  0.004831876
Soft_drinks         0.232244140  0.555124311 -0.16942648  0.103508492
Alcoholic_drinks   -0.463968168  0.113536523 -0.49858320 -0.316290619
Confectionery      -0.029650201  0.005949921 -0.05232164  0.001847469
```

PCA looks to be a super useful method for gaining some insight into high dimensional data that is difficult to examine n other ways.