Class 7: Machine Learning 1

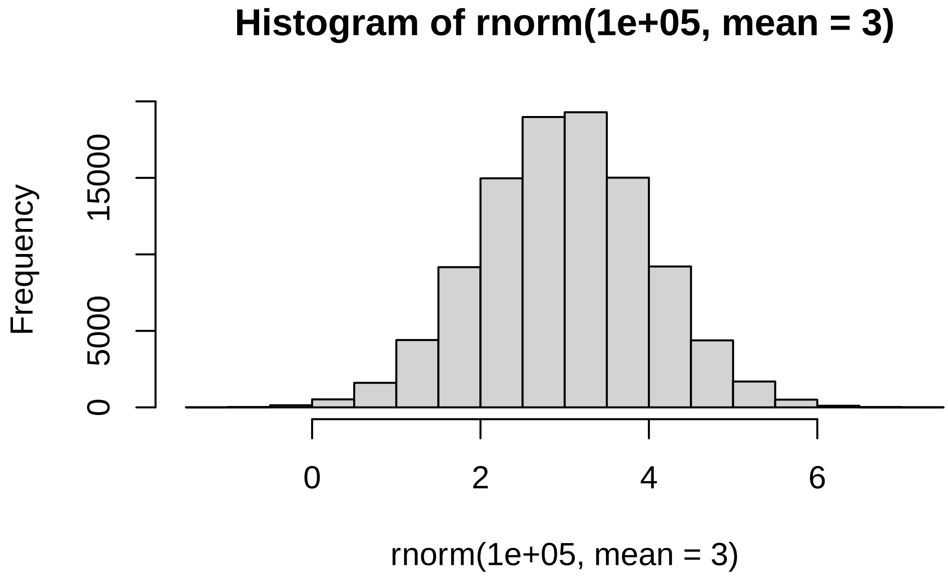
Cynthia Perez (A16393492)

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

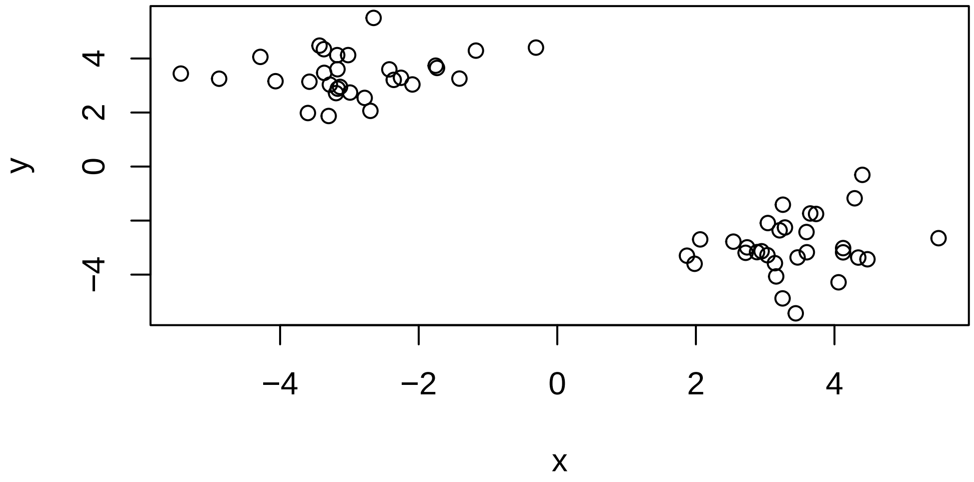
Clustering

Let's start with "k-means" clustering. The main function in base R for this is `kmeans()`

```r
#Make up some data
hist( rnorm(100000, mean=3) )
```



```
#add both rnorm values into one vector
tmp <- c(rnorm(30, -3), rnorm(30, +3))
# first 30 values start at -3 last 30 values are above 3
x <- cbind(x=tmp, y=rev(tmp))
plot(x)
```



Now that we have input data lets try `kmeans()`

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

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

Cluster means:
          x         y
1 -2.936049  3.397500
2  3.397500 -2.936049

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] 50.95272 50.95272
 (between_SS / total_SS =  92.2 %)

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?

```
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
```

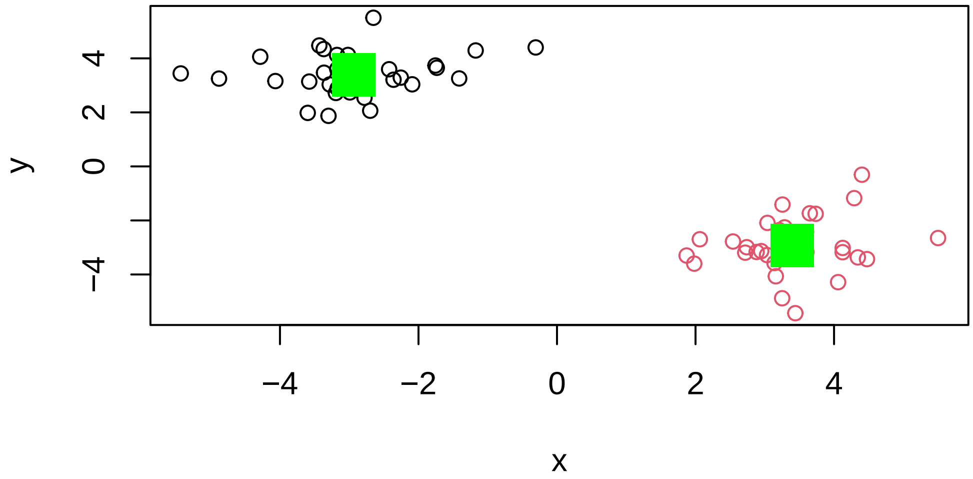
Q. What are the centers/mean values of each cluster?

```
km$centers
```

```
          x         y
1 -2.936049  3.397500
2  3.397500 -2.936049
```

Q. Make a plot of your data showing your clustering results?

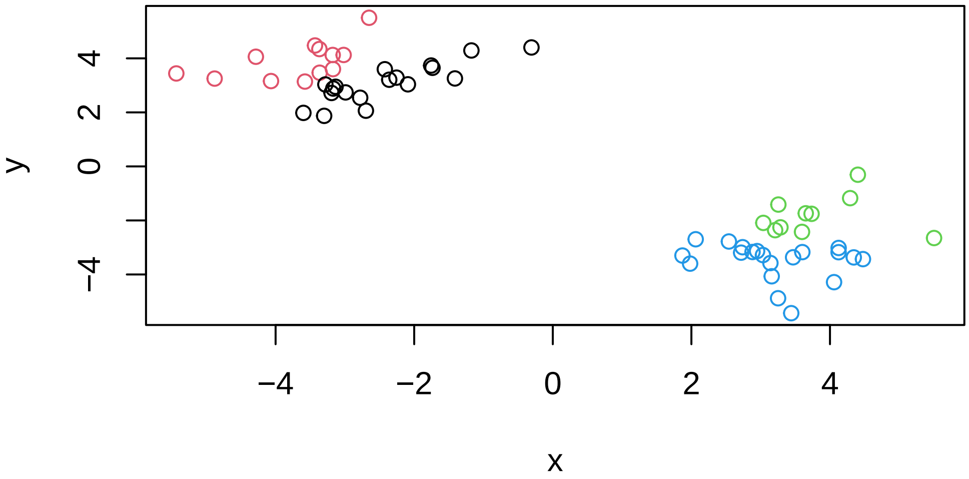
```
#color by cluster using `km$cluster` vector
plot(x, col=km$cluster)
#color the cluster centers
points(km$centers, col="green", pch=15, cex=3)
```



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

```
# kmeans of vector x with 4 groups
km4 <- kmeans(x, centers = 4)

#plot km4
plot(x, col=km4$cluster)
```



Hiearchical Clustering

This form of clustering aims to reveal the structure in your data by progressively grouping points into smaller number of clusters.

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

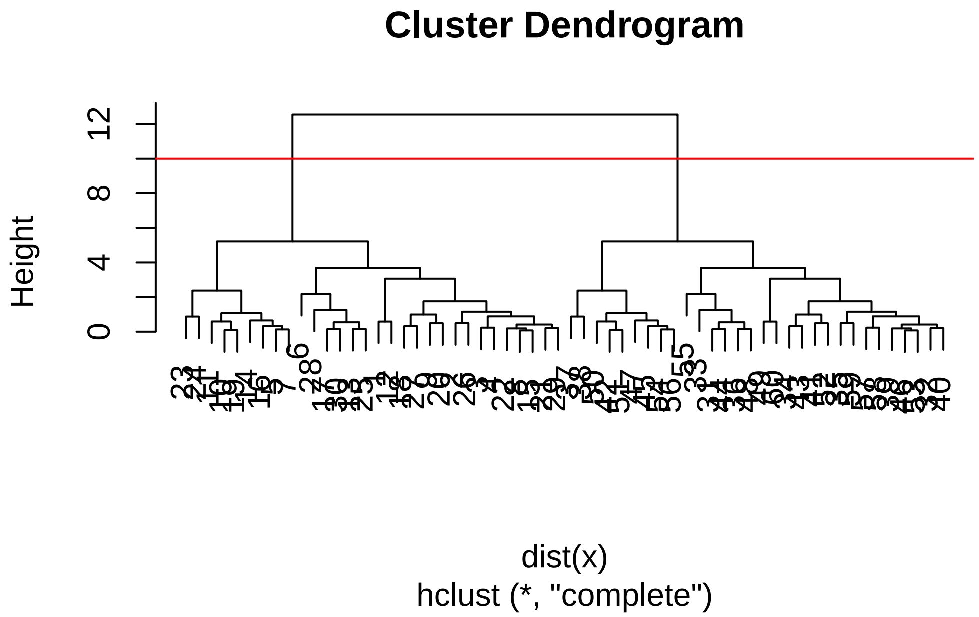
```
# `dist()` measures distance pairwise from point to point
hc <- hclust(dist(x))
hc
```

```
Call:
hclust(d = dist(x))

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

The print out above is not very useful (unlike that from kmeans) but there is a useful `plot()` method.

```
plot(hc)
abline(h=10, col="red")
```



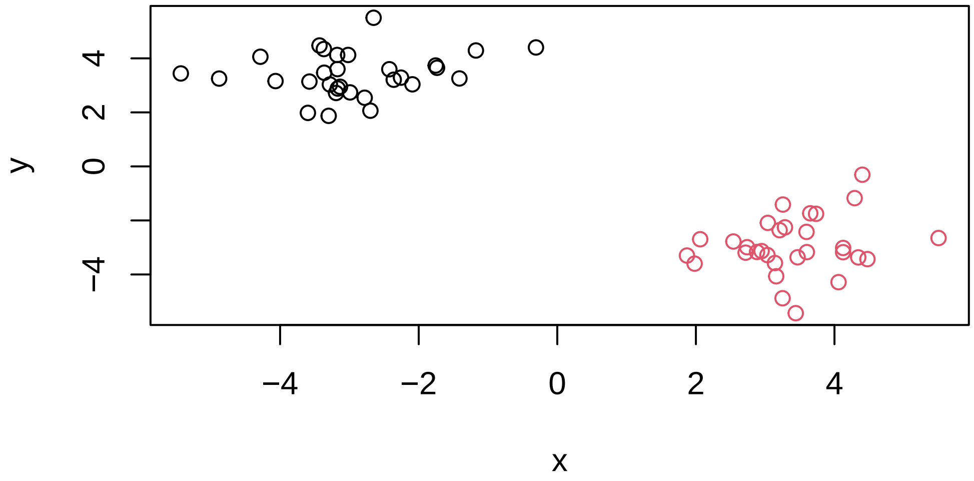
To get my main result (my cluster membership vector) I need to "cut" my tree using `cutree()`

```
#tree cut at height 10 creates 2 groups
grps <- cutree(hc, h=10)
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 hc and color by groups (grps)

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



##PCA of UK Food Data

```r
#read the input file
url <- "https://tinyurl.com/UK-foods"
#use row.names to remove column name (x)
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
```

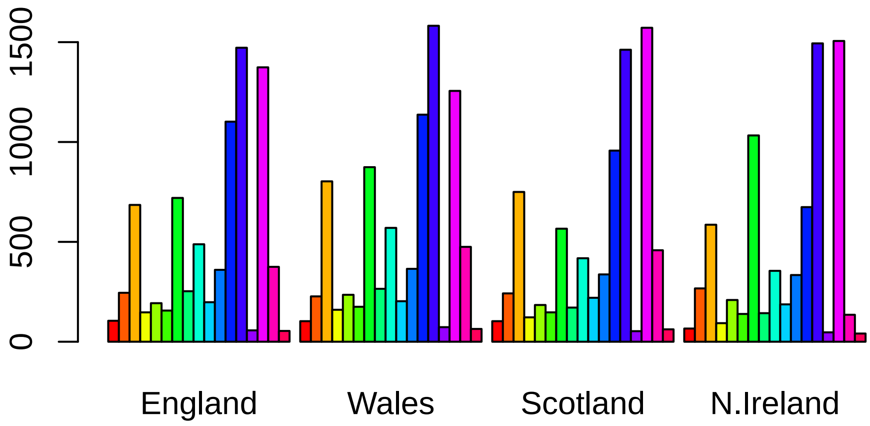
Q. How many rows and columns are in in data frame x?

```
dim(x)
```

```
[1] 17  4
```

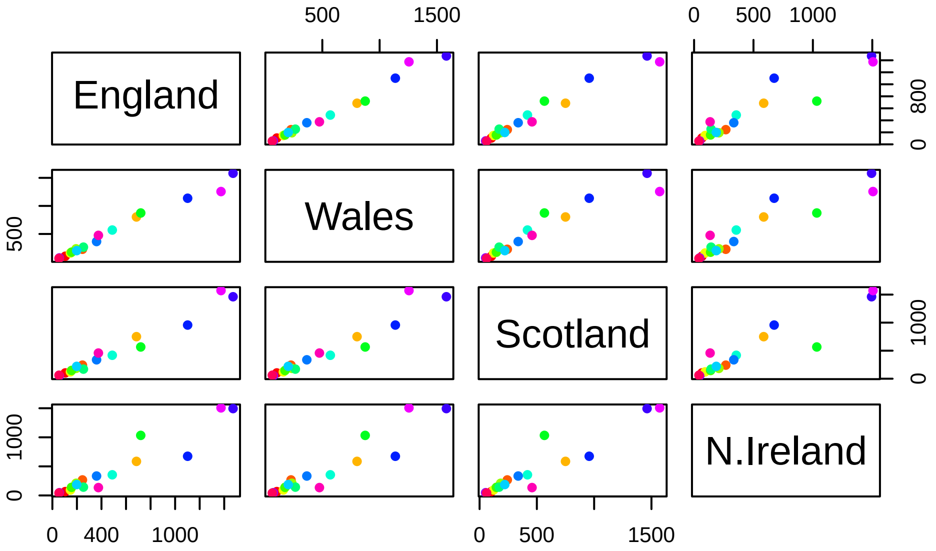
Use barplot to spot trends

```
#change beside to TRUE to unstack the graph
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



We can alternatively use the "pairs" plot for small datasets:

```
#color by number of food rows
#pch= plotting character to visualize the points better
pairs(x, col=rainbow(nrow(x)), pch=16)
```



The pairs plot is useful for small data sets but it can be too much work to interpret and becomes even harder to read with larger data sets.

Use PCA instead with the function `prcomp()`

```
# need to transpose x to perform PCA on the food and not the countries (switch the rows an
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
```

Take a look at what is in pca

```
attributes(pca)
```

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

```
$class
[1] "prcomp"
```

x is what the data looks like on the new axis

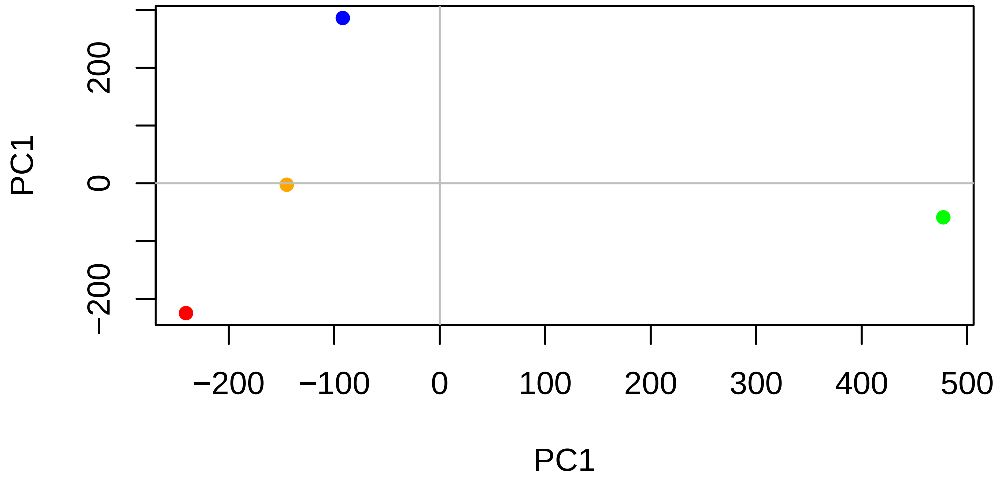
```
pca$x
```

```
                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
```

```
#PC1 captures the most variance (makes it more important)
```

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

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



```
#Ireland sticks out as the green point
```

Another important output from PCA is called the “loadings” vector or the “rotation” component- this tells us how much the original variables (the food 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 is a super useful method fro gaining some insight into high dimensional data that is difficult to interpret in other ways.