Class 7: Machine Learning 1

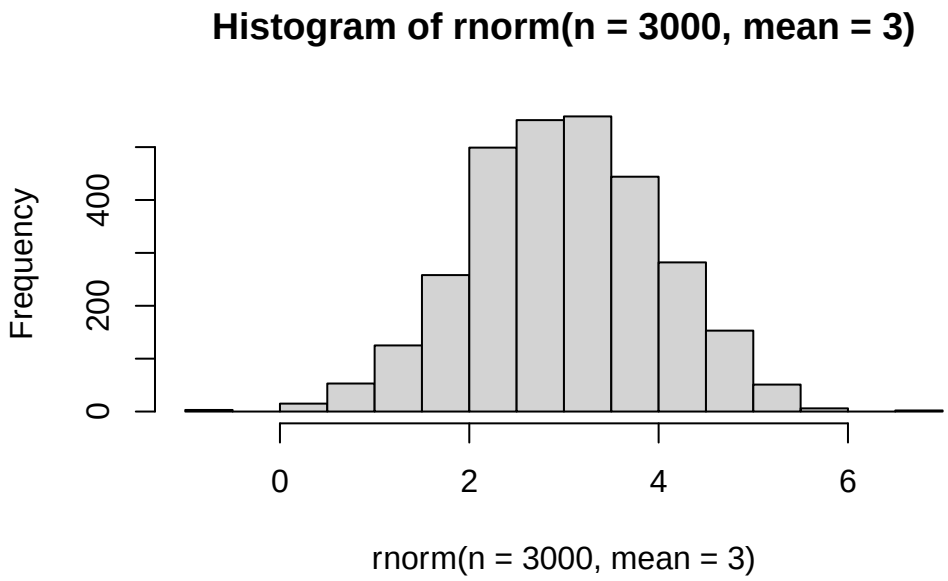
Isabel Philip (A16855684)

Today,we will explore unsupervised machine learning methods including clustering and dimensionality reduction methods.

Let's start by making up some data (where we know there are clear groups) that we can use to test out different clustering methods.

We can use the `rnorm()` function to help us here:

```
hist(rnorm(n=3000, mean= 3))
```

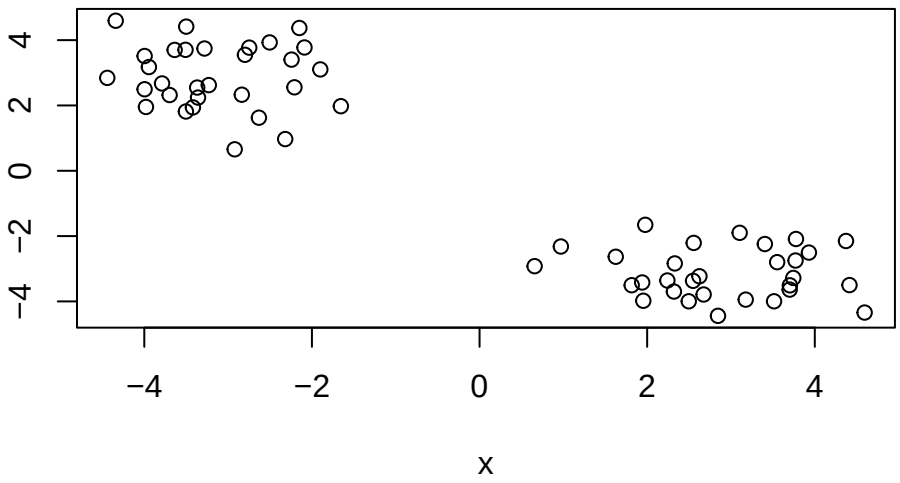


Make data `z` with two "clusters"

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

z <- cbind(x=x, rev(x))
head(z)
```

```
             x
[1,] -4.341206 4.594824
[2,] -2.798980 3.551108
[3,] -3.230404 2.623582
[4,] -2.921385 0.657141
[5,] -3.996477 3.514149
[6,] -3.503775 1.815372
```

```r
plot(z)
```



K-means Clustering

The main function in "base" R for K-means clustering is called `kmeans()`

```r
k <- kmeans(z, centers =2)
k
```

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

Cluster means:
          x
1  2.877513 -3.133490
2 -3.133490  2.877513

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] 45.71942 45.71942
 (between_SS / total_SS =  92.2 %)

Available components:

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

How big is z?

```r
c(nrow(z), ncol(z))
```

```
[1] 60  2
```

```r
attributes(k)
```

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

$class
[1] "kmeans"
```

Q. How many points lie in each cluster?

```
k$size
```

```
[1] 30 30
```

Q. What component of our results tells us about the cluster membershop (i.e. which point lies in which cluster?)

```
k$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
```

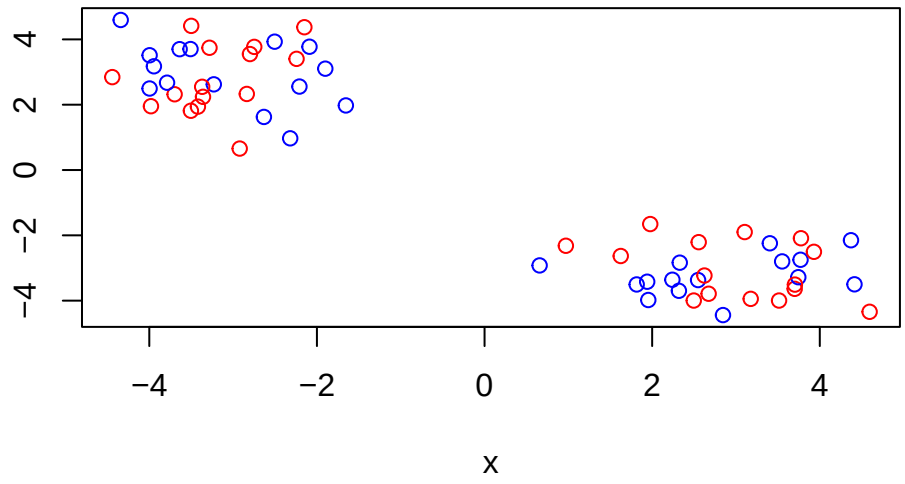
Q. Center of each cluster?

```
k$centers
```

```
          x
1  2.877513 -3.133490
2 -3.133490  2.877513
```

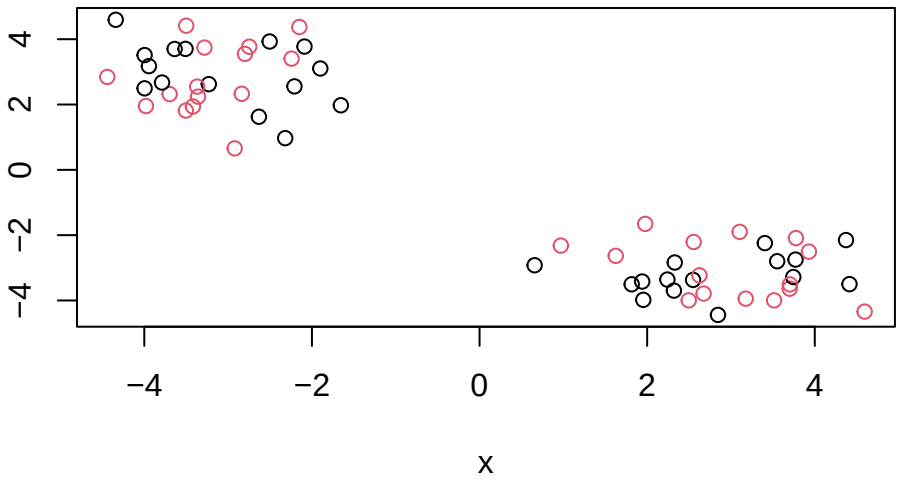
Q. Put this result info together and make a little "base R" plot of our clustering. Also add the cluster center points to this plot.

```
plot(z, col = c("blue", "red"))
```



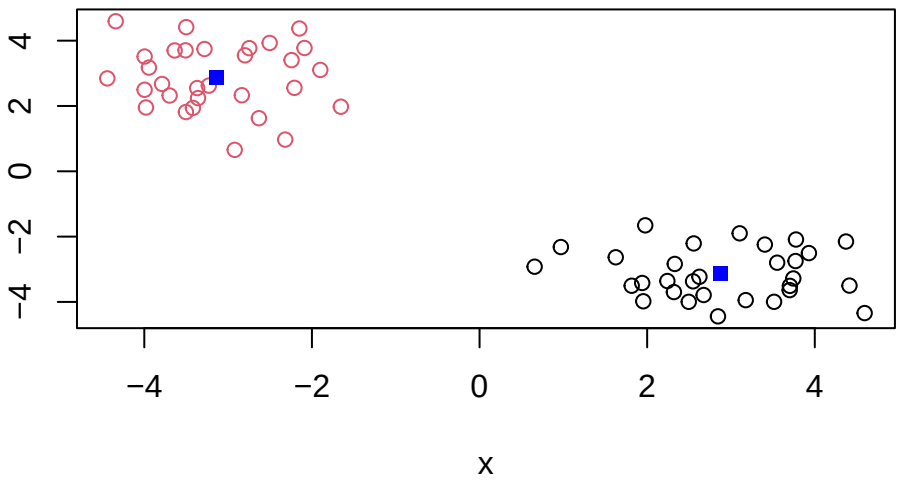
You can color by number

```
plot(z, col = c(1, 2))
```



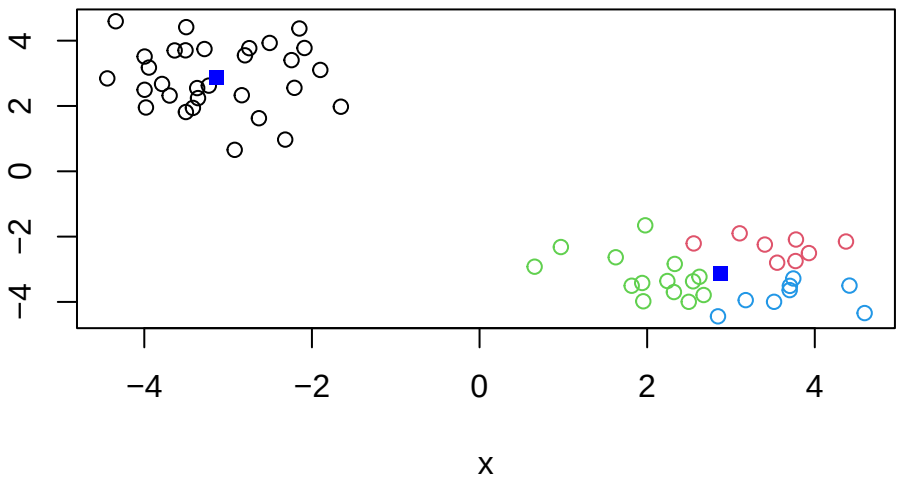
Plot colored by cluster membership:

```
plot(z, col = c(k$cluster))
points(k$centers, col = "blue", pch = 15)
```



Q. Run kmeans on our input **z** and define 4 clusters making the same result vizulation plot as above (plot of z colored by cluster membership)

```
k4 <- kmeans(z, centers =4)
plot(z, col = c(k4$cluster))
points(k$centers, col = "blue", pch = 15)
```




```
k4$totss
```

```
[1] 1175.404
```

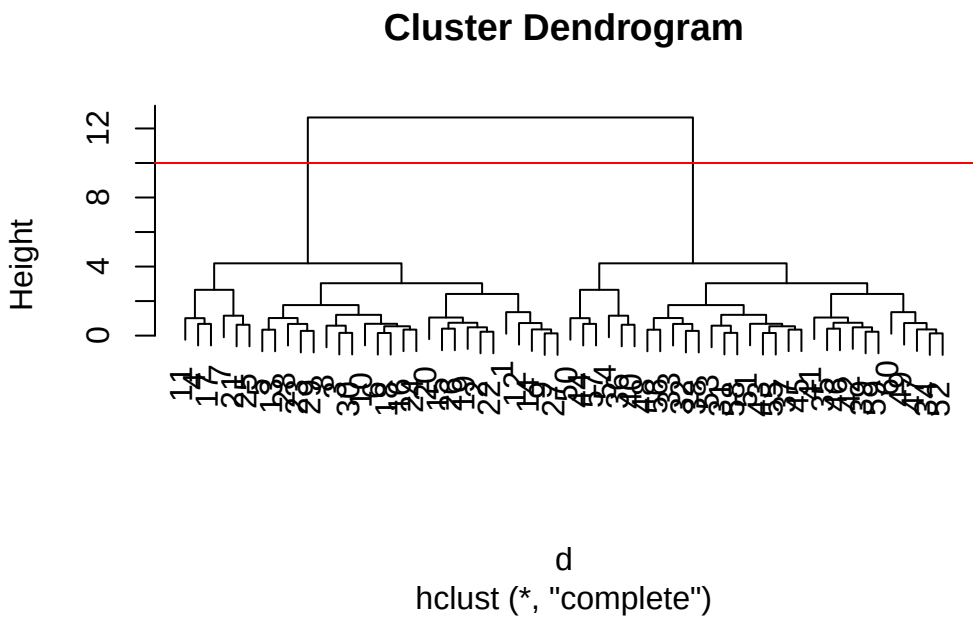
Hierarchical Clustering

The main function in base R for this called `hclust()`. It will take as input a distance matrix (key point is that you can't just give your "raw" data as input - you have to first calculate a distance matrix from your data)

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

```
Call:
hclust(d = d)

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

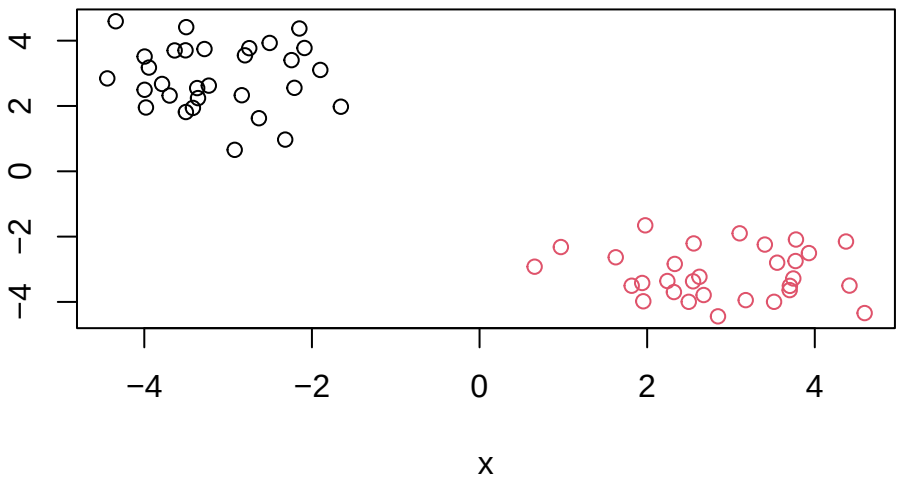
```
plot(hc)
abline(h=10, col = "red")
```



Once I inspect the "tree", I can "cut" the tree to yield my groupings/ clusters. The function that can do this is called `cutree()`

```
grps<- cutree(hc, h=10)
```

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



Hands on with PCA- Principal Component Analysis

Let's examine some silly 17-dimension data detailing food consumption in the UK (England, Scotland, Wales, and N. Ireland).Are these countries eating habits different or similar to each other?

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

```
nrow(x)
```

```
[1] 17
```

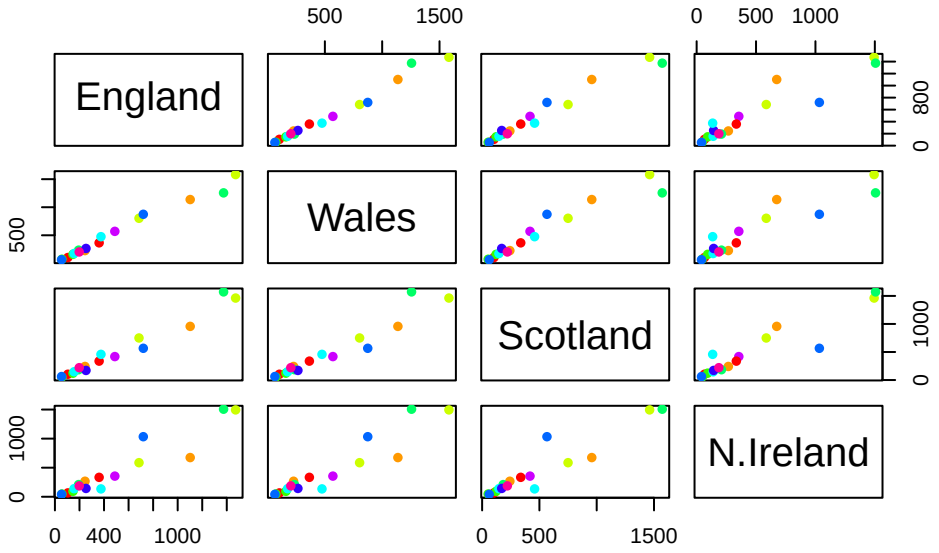
```
ncol(x)
```

```
[1] 4
```

#can also use the dim() function to get the same answer

> 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)
```



Diagonal refers to the Y axis. For the first 3 plots in the first row, is England for example. Plots of all the countries vs. all the countries. If a point lies on a straight line, there are similar amounts of food consumption in both countries. If they aren't, there is more/ less consumed than in the comparing country.

Looking at these types of "pairwise plots" can be helpful but it does not scale well and kind of sucks! There must be a better way..

PCA to the Rescue!

The main function of PCA in base R is called `prcomp()`. This function wants the transpose our input data - i.e. the important food categories in as columns, not rows.

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

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 2.921e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

Let's see what is in our PCA result object `pca`.

```
attributes(pca)
```

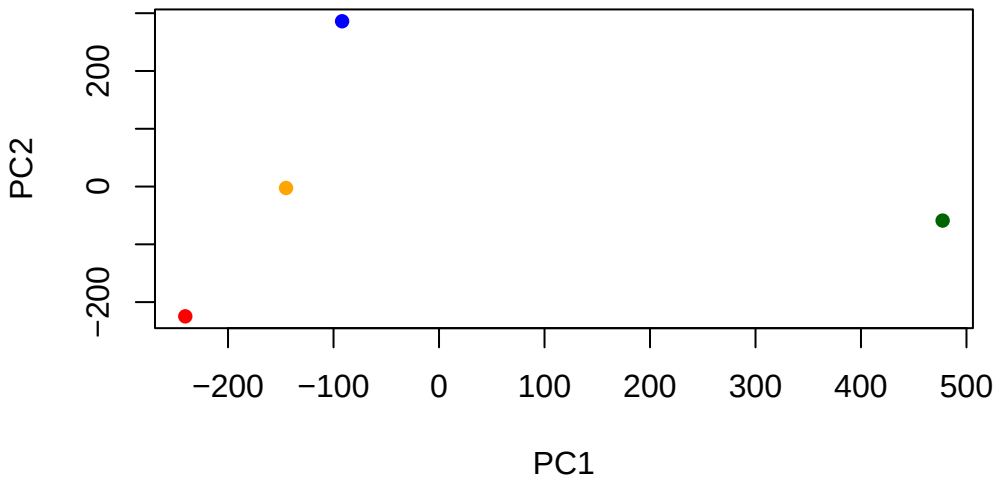
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

The `pca$x` result is where we will focus first as this details how the countries are related to eeach other in terms of our new "axis" (a.k.a "PCs". "eigenvectors", etc.)

```
head(pca$x)
```

```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -9.152022e-15
Wales     -240.52915 -224.646925 -56.475555  5.560040e-13
Scotland   -91.86934  286.081786 -44.415495 -6.638419e-13
N.Ireland  477.39164  -58.901862  -4.877895  1.329771e-13
```

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



We can look at the so-called PC "loadings" result object to see how the original foods contribute to our new PCs (i.e. how the original variables contribute to our new better PC varaibles)

```
pca$rotation[,1]
```

```
          Cheese       Carcass_meat         Other_meat                Fish
    -0.056955380         0.047927628       -0.258916658       -0.084414983
Fats_and_oils              Sugars     Fresh_potatoes          Fresh_Veg
    -0.005193623       -0.037620983        0.401402060       -0.151849942
   Other_Veg  Processed_potatoes       Processed_Veg         Fresh_fruit
    -0.243593729       -0.026886233       -0.036488269       -0.632640898
      Cereals            Beverages       Soft_drinks    Alcoholic_drinks
    -0.047702858       -0.026187756        0.232244140       -0.463968168
Confectionery
    -0.029650201
```

[,1] is PC1

Digging deeper (variable loadings)

```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```