class07

Mina Wu (A59013200)

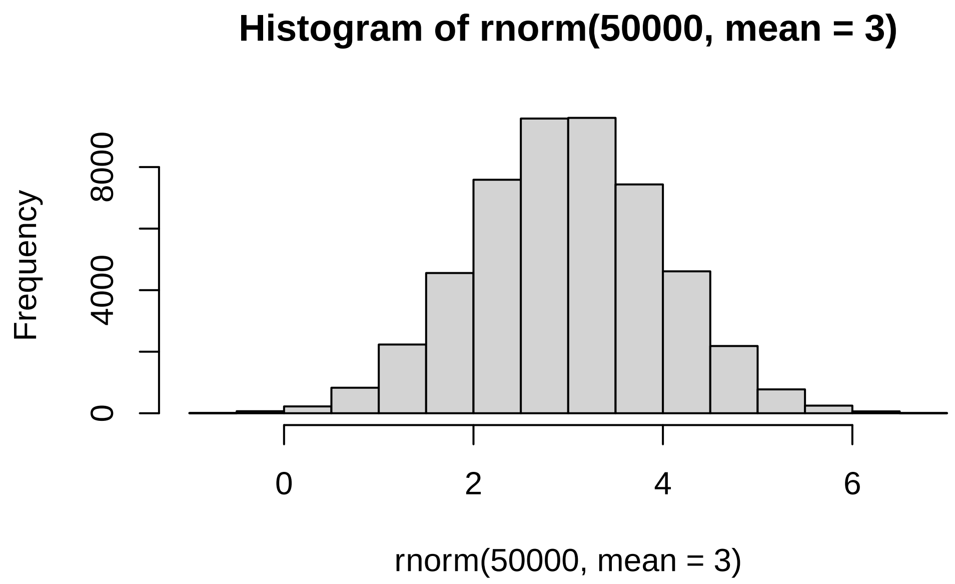
2024-01-31

Today we are going to explore some core machine learning methods. Namely clustering and dimensionality reduction approaches.

Kmeans clustering

The main function for k-means in "base" R is called `kmeans()`. Let's first make up some data to see how kmeans works and to get at the results.

```r
hist(rnorm(50000, mean=3))
```



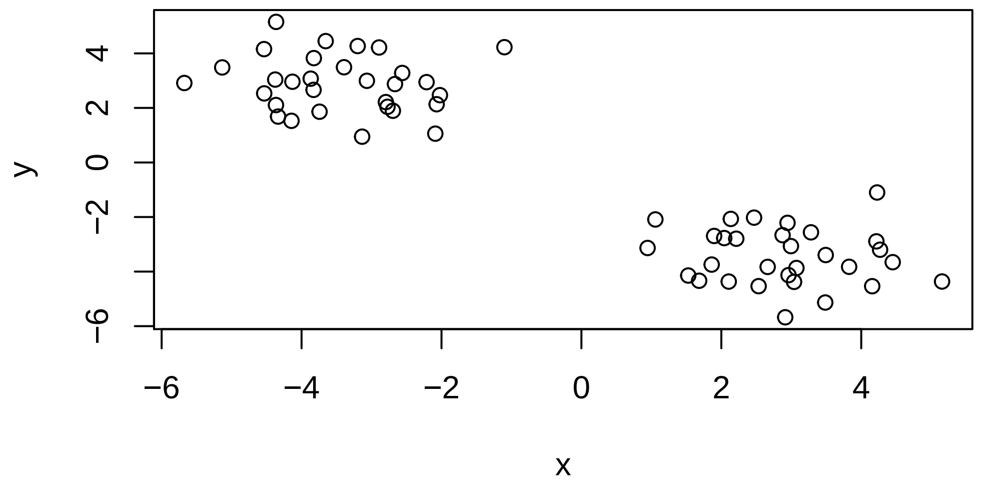
Make a wee vector with 60 total points half centered at +3 and half centered at -3.

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

```
 [1]  2.9462402  1.6831047  2.1060236  2.0414446  4.4500104  2.9133869
 [7]  1.8971422  2.9606544  1.8619032  3.4854480  1.0572305  2.9949316
[13]  3.8269230  4.1566482  0.9460158  3.2814959  5.1550833  4.2691873
[19]  2.8757692  2.6623884  4.2165154  2.2137103  1.5276517  2.4680147
[25]  4.2266682  3.4923890  2.1357236  3.0734172  2.5332365  3.0394980
[31] -4.3766072 -4.5340105 -3.8693593 -2.0692917 -3.3933618 -1.1001413
[37] -2.0221615 -4.1444268 -2.7951515 -2.8920546 -3.8284064 -2.6651104
[43] -3.1982104 -4.3640358 -2.5610630 -3.1346082 -4.5362506 -3.8248224
[49] -3.0666335 -2.0881991 -5.1340434 -3.7430728 -4.1304933 -2.6946165
[55] -5.6758845 -3.6552423 -2.7716627 -4.3657064 -4.3356361 -2.2133748
```

reverse the order of tmp using rev() for y. Then plot using plot().

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



Run `kmeans()` asking for two clusters: and nstart=20, which is the iterations

```r
k <- kmeans(x, centers=2, nstart=20)
k
```

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

Cluster means:
          x         y
1  2.883262 -3.439455
2 -3.439455  2.883262

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] 62.98465 62.98465
 (between_SS / total_SS =  90.5 %)

Available components:

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

What is in this result objects?

```r
attributes(k)
```

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

$class
[1] "kmeans"
```

What are the cluster centers?

```r
k$centers
```

```
          x         y
1  2.883262 -3.439455
2 -3.439455  2.883262
```

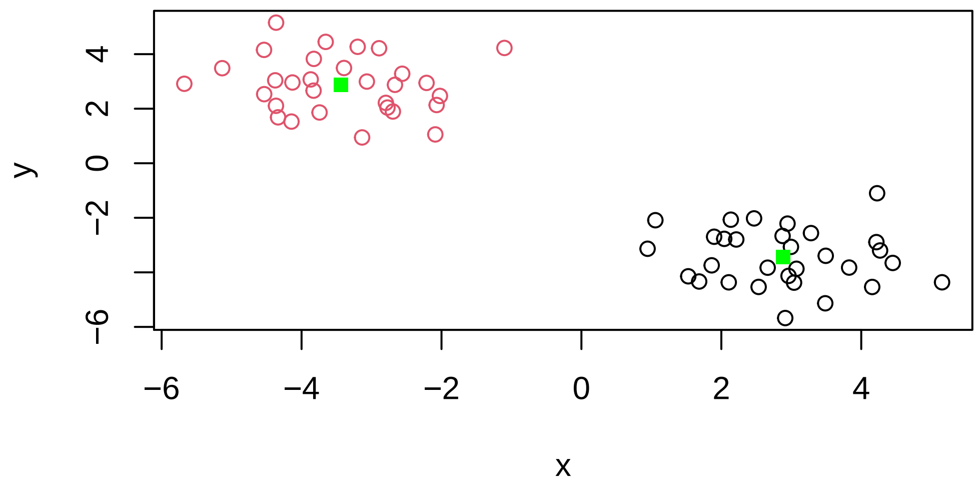
What is my clustering resluts? I.E. what cluster does each points reside in?

```
k$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
```

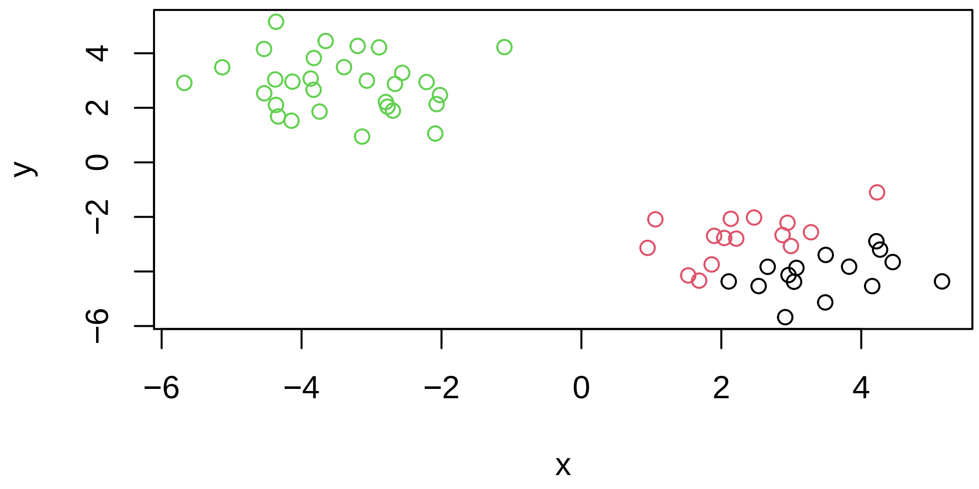
> Q. Plot your data `x` showing your clusterng result and the center point for each cluster?

```
plot(x, col= k$cluster)
points(k$centers, pch=15, col ="green")
```



> Q.Run kmeans and cluster into 3 groups and plot the result?

```
k3 <- kmeans(x, centers=3)
plot(x, col=k3$cluster)
```



```
k$tot.withinss
```

```
[1] 125.9693
```

```
k3$tot.withinss
```

```
[1] 101.1119
```

The big limitation of kmeans is that it imposes a structure on our data (i.e. a clustering) that you ask for in the first place. K-means will always give you the renumber of clusters you request.

hierarchical clustering

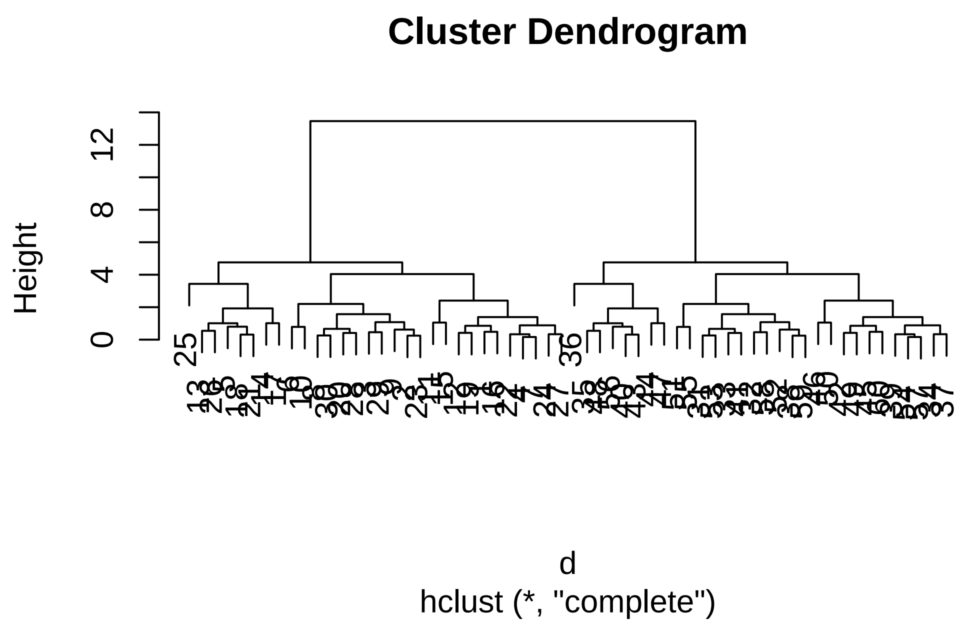
The main function in"base" R for this is called 'hlcust()'. It wants a distance matrix as input no the data itself.

We can calculate a distance matrix in lots of different ways but here we will use the 'dist()' function.

```r
d <- dist(x, diag = T)
hc<- hclust(d)
```

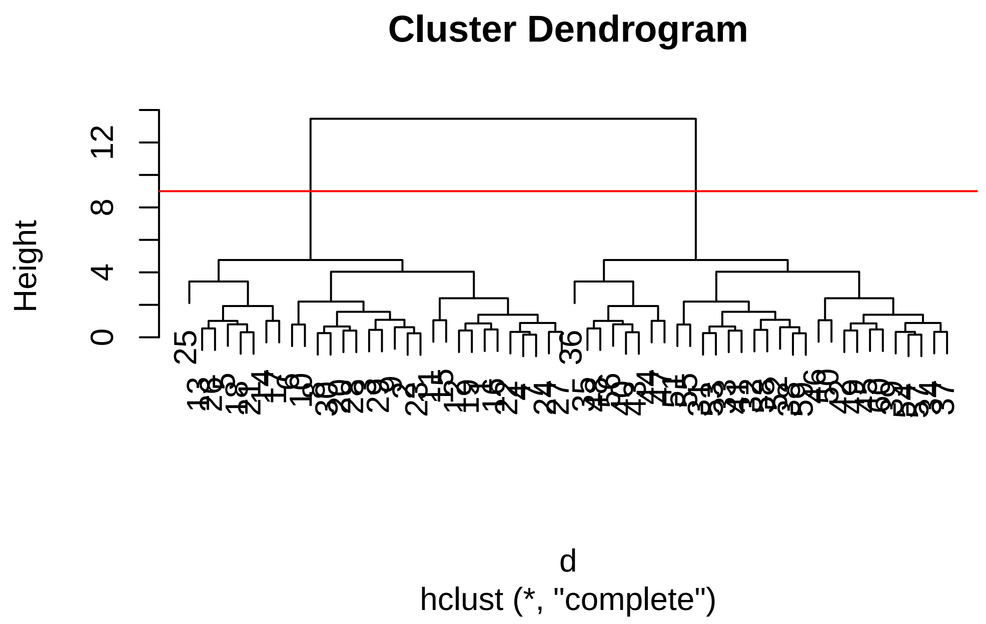
There is a specific plot method for hclust objecs. Let's see it:

```r
plot(hc)
```



cut the cluster at height 9 using abline.

```r
plot(hc)
abline(h=9, col="red")
```



To get the cluster membership vector we need to "cut" the tree at a given height that we pick. The function to do this is called `cutree()`

```
cutree(hc, h=9)
```

```
 [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
```

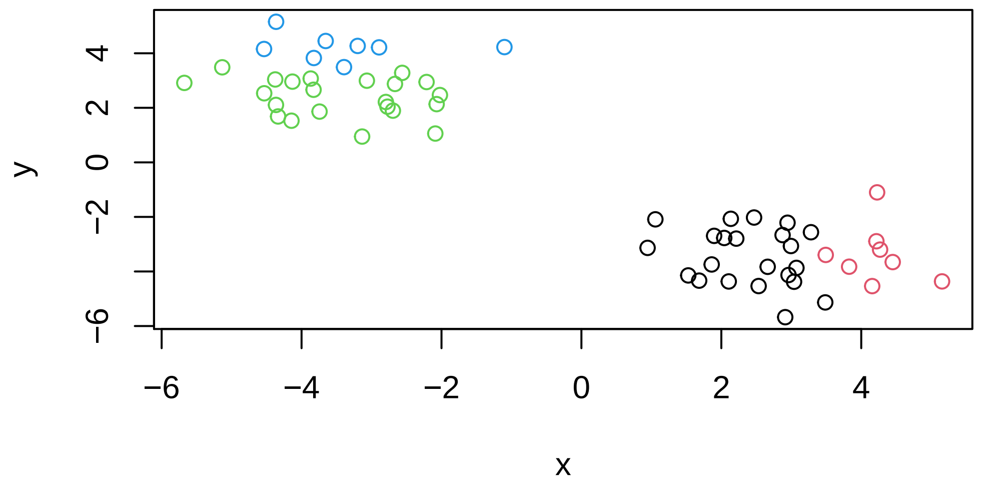
Use k to cut into 4 clusters.

```
grps<- cutree(hc, k=4)
grps
```

```
 [1] 1 1 1 1 2 1 1 1 1 1 1 1 2 2 1 1 2 2 1 1 2 1 1 1 2 2 1 1 1 1 3 3 3 3 4 4 3 3
[39] 3 4 3 3 4 4 3 3 4 4 3 3 3 3 3 3 3 4 3 3 3 3
```

Q. Plot our data (x) colored by our hclust result.

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



Principal Component Analysis (PCA)

We will start with PCR of a tiny tiny dataset and make fun of stuff Barry eats.

```
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
```

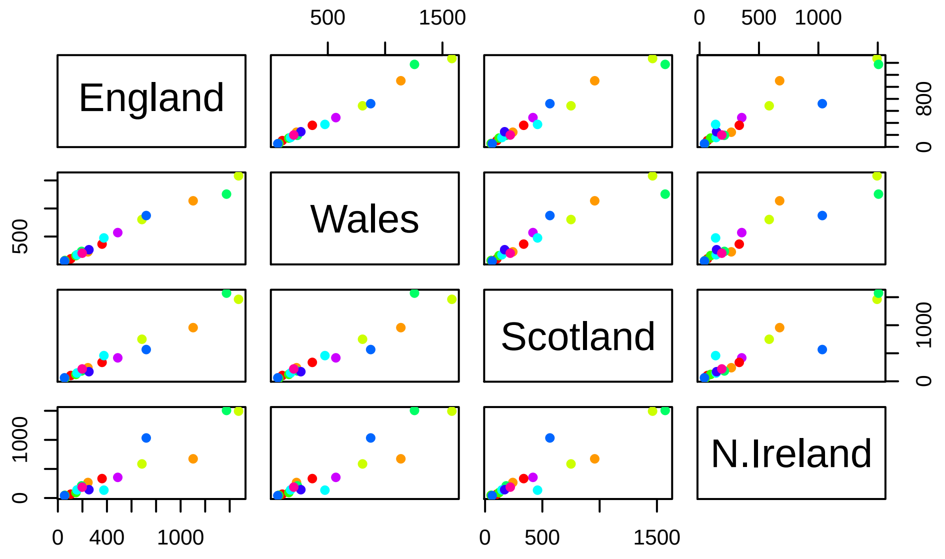
```
dim(x)
```

```
[1] 17  4
```

```
# there are 17 rows and 4 columns.
```

One useful plot in this case (because we only have 4 countries to look across) is a so-called pairs plot.

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



Enter PCA

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

It wants our foods as the columns and the countries as the rows. It basically wants the transpose of the data we have.

```r
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
```

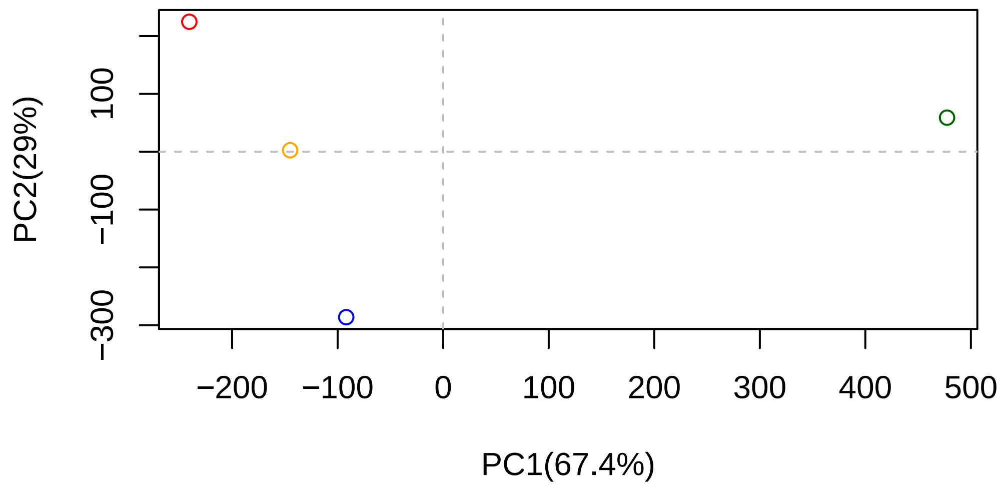
```r
attributes(pca)
```

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

$class
[1] "prcomp"
```

Plot the PCA plot

```r
plot(pca$x[,1], pca$x[,2], xlab= "PC1(67.4%)", ylab= "PC2(29%)", col= c("orange", "red", "
abline(h=0, col="gray", lty=2)
abline(v=0, col="gray", lty=2)
```



PC2(29%)
100
−100
−300
−200
−100
0
100
200
300
400
500
PC1(67.4%)