Lab 7 Machine Learning 1

Thoi Tran (A17035545)

Today we are going to laern how to apply different machine learning methods begining with clustering:

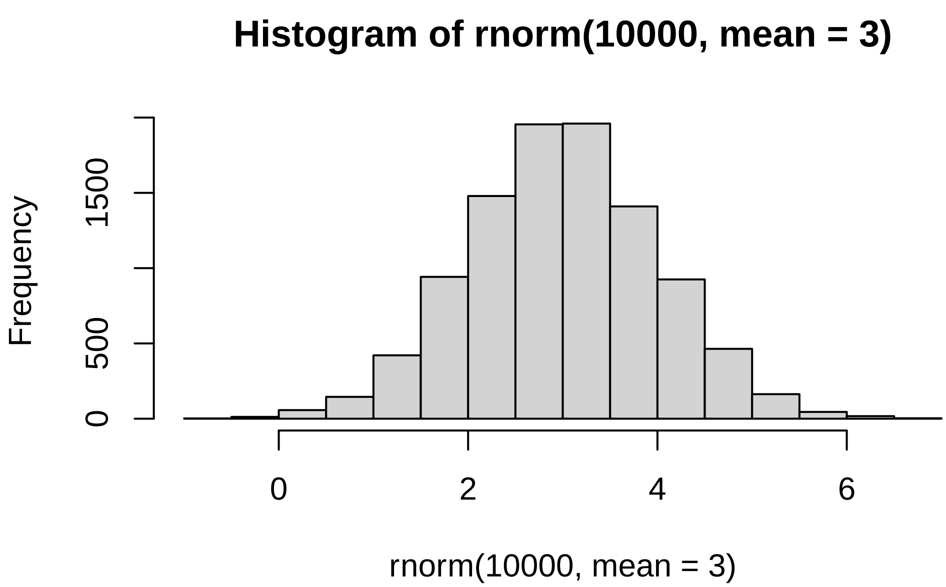
The goal hear is to find group/clusters in your input data.

First I will make up some data with clear groups. For this I will use the `rnorm()` function.

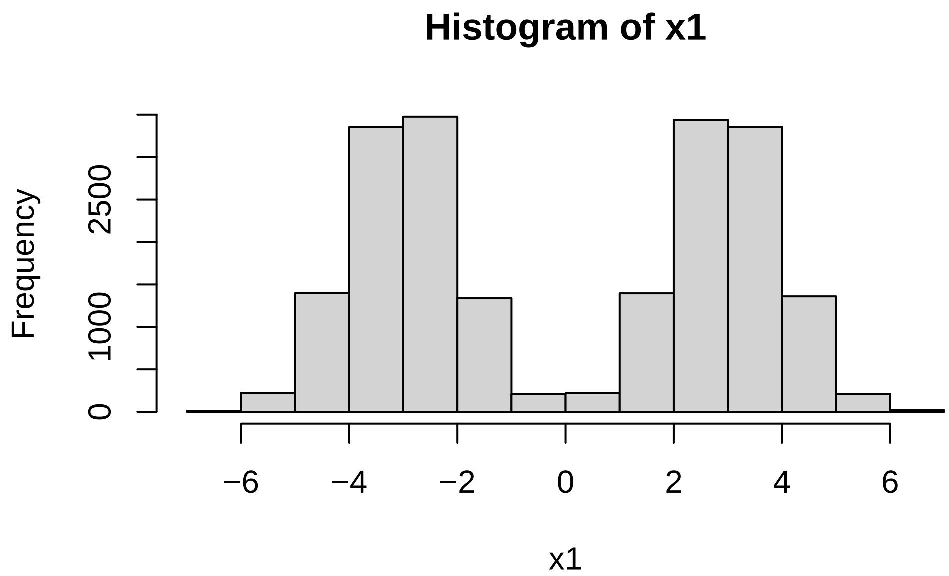
```
rnorm(10)
```

```
 [1]  0.04029517 -1.13658746  0.91907931  0.26975839 -1.72068771 -1.58037588
 [7]  1.27724954  0.77822301  0.85095336  0.83982910
```

```
hist(rnorm(10000, mean = 3))
```



```r
x1 <- c(rnorm(10000, mean = 3),
        rnorm(10000, mean = -3))
hist(x1)
```




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

```
 [1]  2.3768350  2.9973342  2.1842681  1.6847966  3.1513466  2.4196309
 [7]  2.7152919  4.3935185  1.7428376  3.0174235  1.5159110  3.2200436
[13]  2.8960347  2.3559274  3.9095316  4.5949817  3.6737325  2.7482953
[19]  2.0404963  1.2657200  3.0884499  3.4294076  3.1764566  2.1219954
[25]  5.3136093  2.6428071  0.8516846  1.4160197  2.7002378  4.0210024
[31] -3.2208292 -4.3039351 -2.5446458 -1.9835657 -4.3624589 -3.2270507
[37] -2.1402619 -2.7537926 -1.7354415 -3.3719119 -2.1478115 -2.3539349
[43] -3.0914865 -3.2919338 -2.9899810 -3.4128246 -1.8160872 -3.5548561
[49] -3.6686480 -3.6754627 -4.8060391 -3.6178336 -3.1185065 -3.0816384
[55] -2.4309373 -2.6788785 -2.4847508 -3.7484769 -3.8243966 -2.8259836
```

```r
y <- rev(x)
y
```

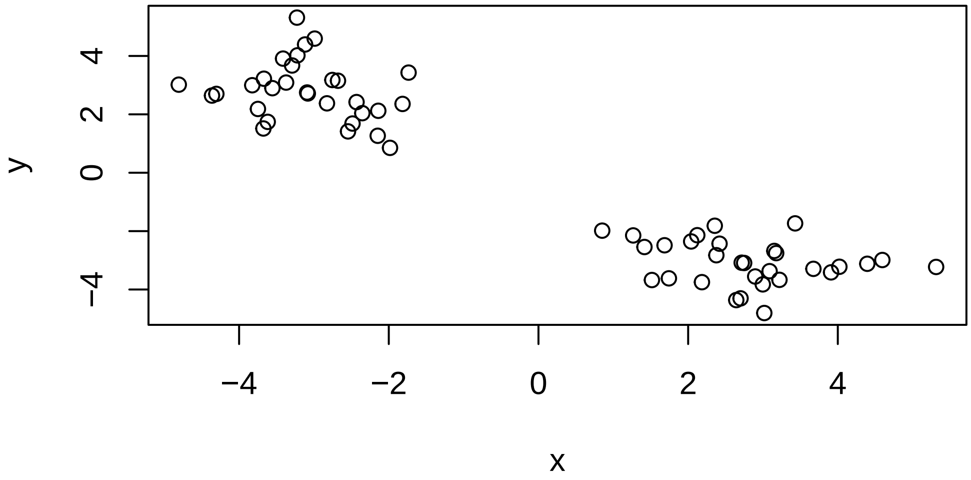
```
 [1] -2.8259836 -3.8243966 -3.7484769 -2.4847508 -2.6788785 -2.4309373
 [7] -3.0816384 -3.1185065 -3.6178336 -4.8060391 -3.6754627 -3.6686480
```

```
[13] -3.5548561 -1.8160872 -3.4128246 -2.9899810 -3.2919338 -3.0914865
[19] -2.3539349 -2.1478115 -3.3719119 -1.7354415 -2.7537926 -2.1402619
[25] -3.2270507 -4.3624589 -1.9835657 -2.5446458 -4.3039351 -3.2208292
[31]  4.0210024  2.7002378  1.4160197  0.8516846  2.6428071  5.3136093
[37]  2.1219954  3.1764566  3.4294076  3.0884499  1.2657200  2.0404963
[43]  2.7482953  3.6737325  4.5949817  3.9095316  2.3559274  2.8960347
[49]  3.2200436  1.5159110  3.0174235  1.7428376  4.3935185  2.7152919
[55]  2.4196309  3.1513466  1.6847966  2.1842681  2.9973342  2.3768350
```

```r
z <- cbind(x, y)
head(z)
```

```
            x         y
[1,] 2.376835 -2.825984
[2,] 2.997334 -3.824397
[3,] 2.184268 -3.748477
[4,] 1.684797 -2.484751
[5,] 3.151347 -2.678878
[6,] 2.419631 -2.430937
```

```r
plot(z)
```



K_means Clustering

Use `kmeans()` funciton setting k to 2 and nstart = 10

Inspect/print the results

> Q. How many points are in each cluster?

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

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

Cluster means:
          x         y
1  2.788854 -3.075479
2 -3.075479  2.788854

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] 47.5451 47.5451
 (between_SS / total_SS =  91.6 %)

Available components:

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

Results in kmeans object `km`.

```r
attributes(km)
```

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

$class
[1] "kmeans"
```

> Q. What component of your result object details? - Cluster size? - Cluster assignment/membership? - Cluster center?

Cluster size?

```r
km$size
```

```
[1] 30 30
```

Cluster assignment/membership?

```r
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
```

Cluster center?

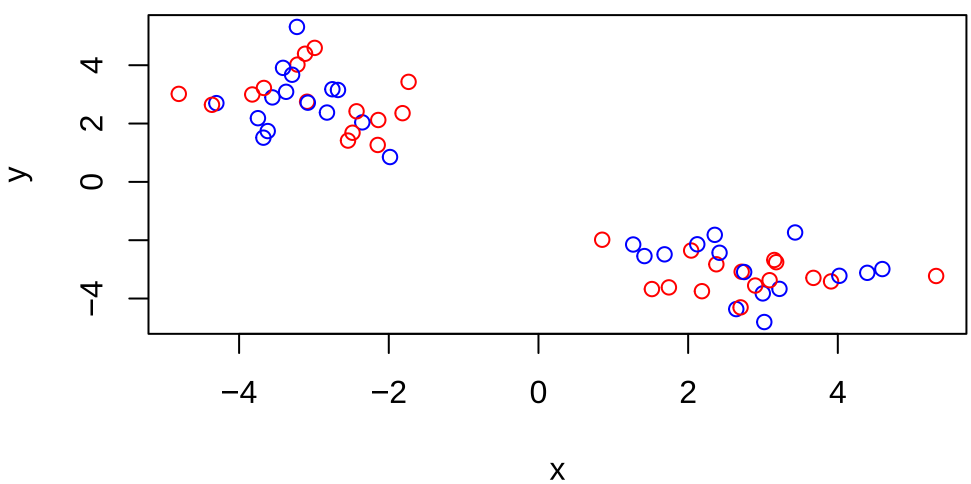
```r
km$centers
```

```
          x         y
1  2.788854 -3.075479
2 -3.075479  2.788854
```

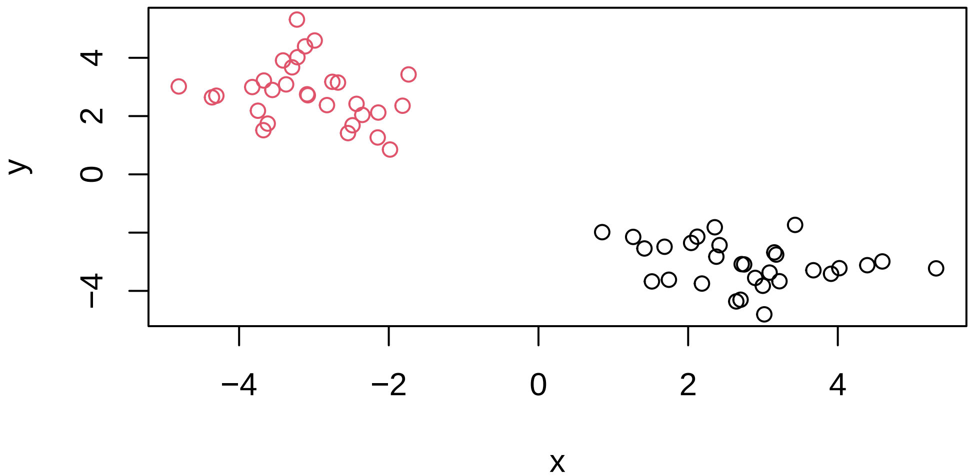
> Q. Plot z colored by the kmeans cluster assignment and add cluster centers as blue points.

R will recycle shorter color vectore to be the same length as the longer (number of data points) in z.

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

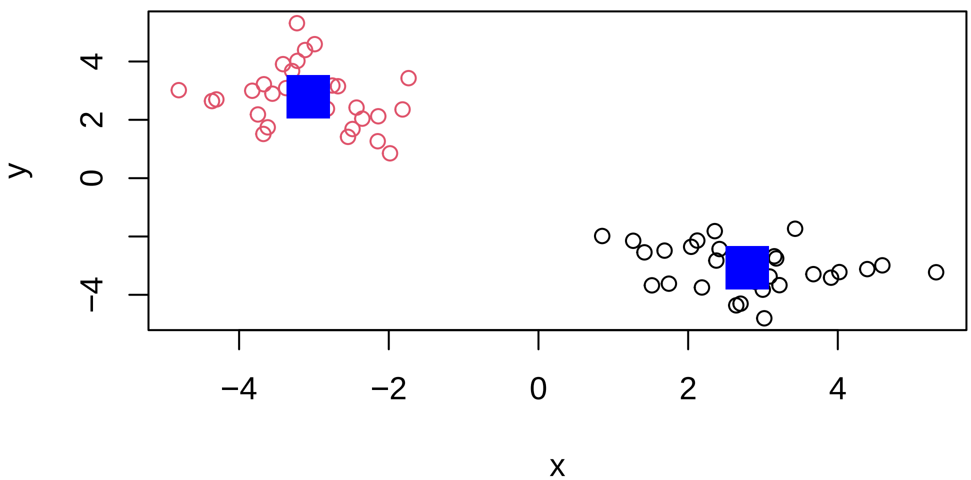


```r
plot(z, col = km$cluster)
```



We can use the `points()` funciton to add new points to an exsiting plot, like the cluster centers.

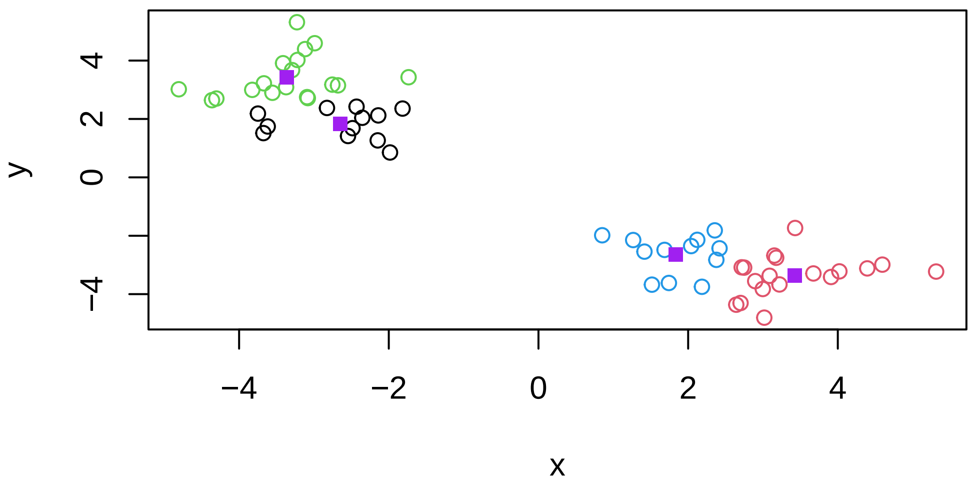
```r
plot(z, col = km$cluster)
points(km$centers, col = "blue", pch = 15, cex = 3)
```



Q. Can you run kmeans and ask for 4 clusters and plot the results?

```r
km_1 <- kmeans(z, centers = 4)
```

```r
plot(z, col = km_1$cluster)
points(km_1$centers, col = "purple", pch = 15)
```



Hierarchical Clustering

Let's take our same data `z` and see how `hclust` works.

First we need a distance matrix of our data to be clustered

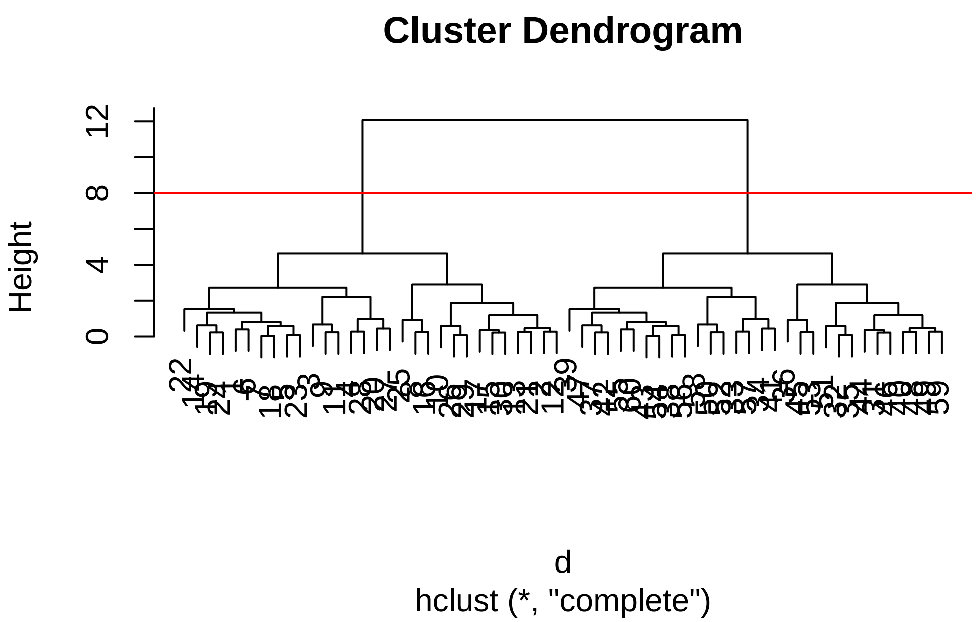
```r
d <- dist(z)
hc <- hclust(d)
hc
```

```

Call:
hclust(d = d)

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

```r
plot(hc)
abline(h = 8, col = "red")
```



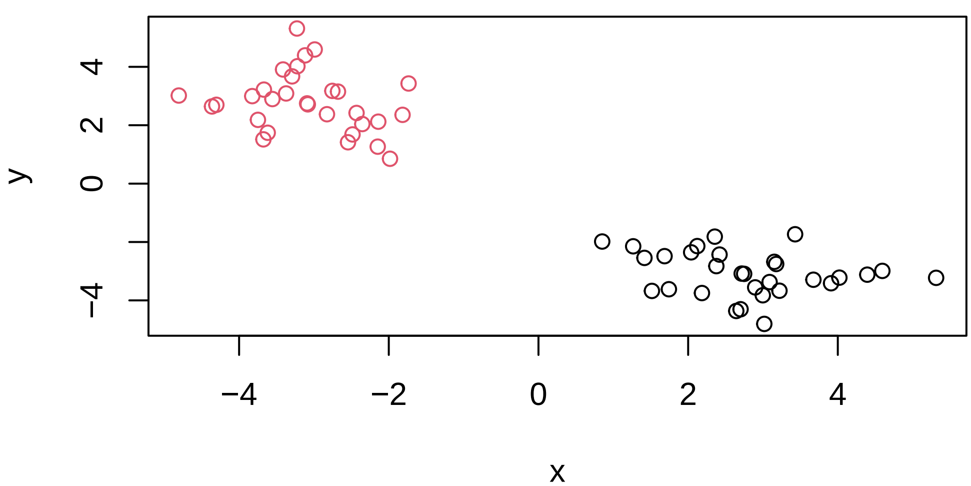
I can get my cluster membership vector by "cutting the tree" with the `cutree()` function.

```
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 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. Can you plot z colored by our hclust results?

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



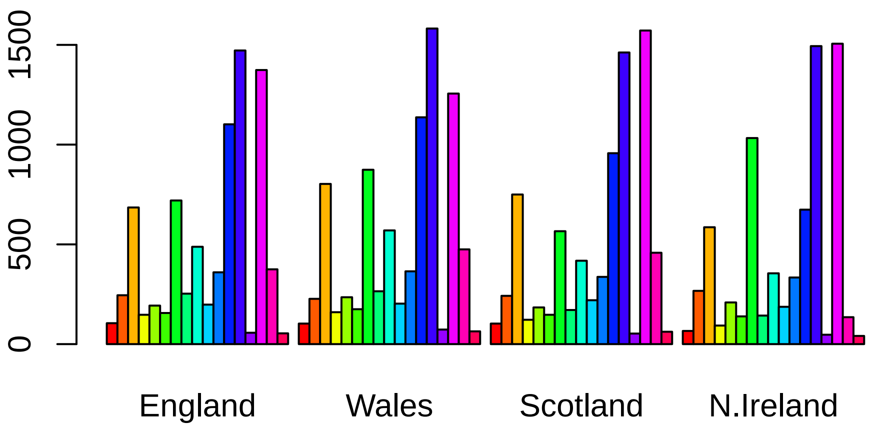
PCA of UK food Data

Read data from the UK on food consumption in different parts of the UK.

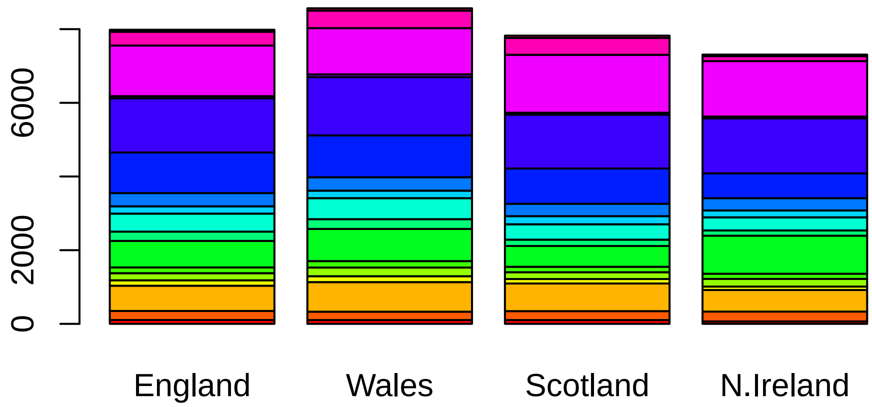
```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names = 1)
head(x)
```

```
               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
```

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

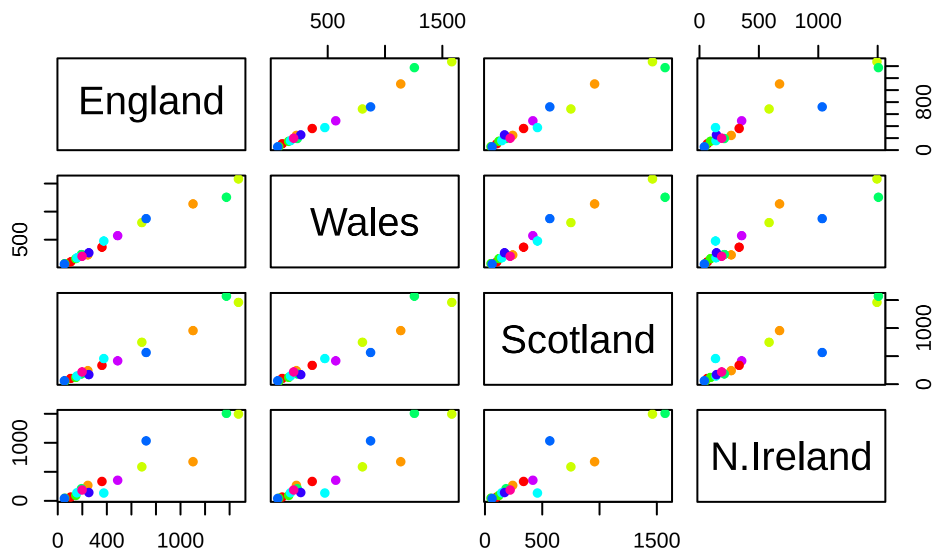



```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



A so-called "Pairs" plot can be usefull for small datasets like this.

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



It's hard to see structure and trends in even this small dataset. How will we ever do this when we have big datasets with 1,000s or 10s of thousands of things we are measuring.

PCA to the rescue

Let's see how PCA deals with this dataset. So the main funciton in base R to do PCA is called `prcomp()`.

```
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
```

Let's see what is inside this `pca` objecct that we created from running `prcomp()`.

```r
attributes(pca)
```

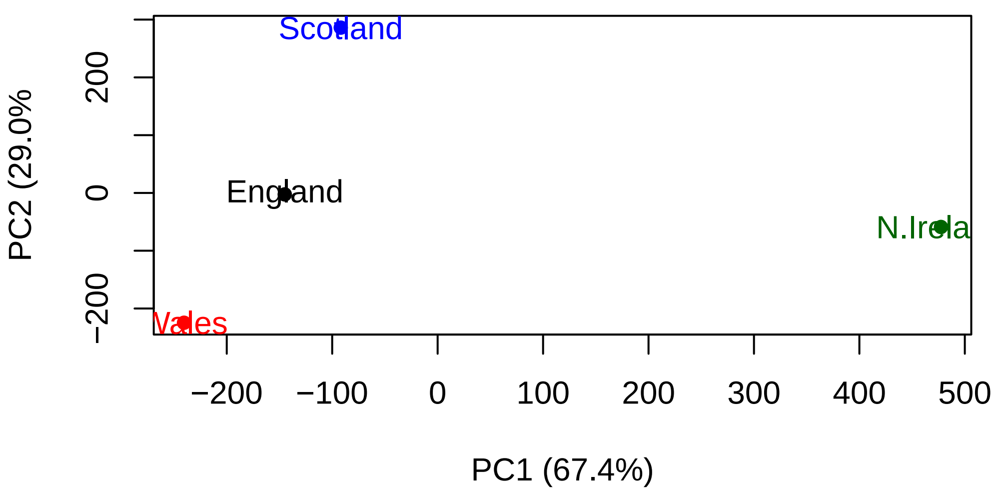
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```r
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
```

```r
plot(pca$x[,1],pca$x[,2], col = c("black", "red", "blue", "darkgreen"),
     pch = 16, xlab = "PC1 (67.4%)", ylab = "PC2 (29.0%")
text(pca$x[,1], pca$x[,2], colnames(x), col = c("black", "red", "blue", "darkgreen"))
```



Digging deeper (variable loadings)

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