class7

Solomon Kim

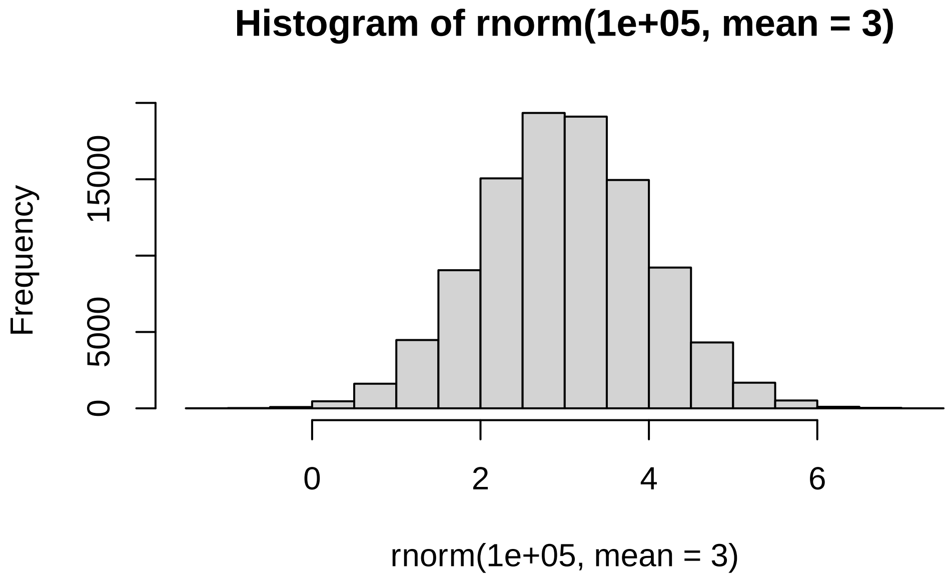
Today we will start out multi-part exploration of some key machine learning methods. We will begin with clustering -finding groupings in data, and then dimensionallity reduction.

Clustering

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

```
# make up some data

hist( rnorm(100000, mean=3) )
```



```r
rnorm(30, -3)
```

```
 [1] -3.5253709 -4.1911867 -2.3236539 -4.0381354 -2.5948770 -1.8026115
 [7] -0.1771642 -2.3657202 -2.8763852 -2.5609678 -3.2633865 -3.1849022
[13] -4.3514637 -3.6744018 -5.1755442 -4.0346041 -3.0841866 -0.6720690
[19] -1.8720779 -1.8651832 -3.7041767 -4.2841026 -2.4523506 -5.4119644
[25] -4.2318123 -3.1344692 -2.2263627 -5.0129588 -2.3497671 -2.3944591
```

```r
rnorm(30, +3)
```

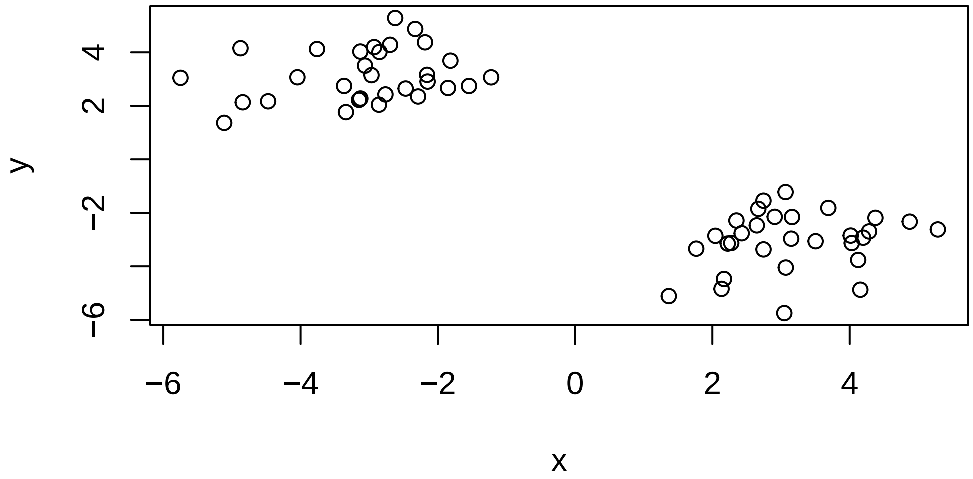
```
 [1] 3.675380 1.813456 3.133407 3.137678 3.022814 2.260704 3.494673 3.027600
 [9] 3.352242 4.039919 3.689758 1.901329 2.736964 2.293465 2.431542 2.266578
[17] 2.531793 3.753981 3.337352 3.028434 2.108416 2.171207 2.241444 2.938005
[25] 4.366440 2.235075 2.298975 2.253501 3.924807 2.173234
```

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

```
 [1] -4.045675 -3.129726 -3.061023 -2.329741 -3.366748 -5.112636 -2.763120
 [8] -5.749494 -2.468983 -1.816312 -2.186873 -2.850432 -1.223489 -3.760501
[15] -3.148678 -2.149858 -3.339314 -4.842681 -4.472793 -4.875170 -2.287930
[22] -2.156973 -1.545977 -1.851721 -2.858137 -2.928317 -3.128303 -2.695909
[29] -2.621467 -2.966490  3.148043  5.285029  4.283303  2.274778  4.196152
[36]  2.043745  2.669548  2.744413  3.160241  2.350524  4.155234  2.169401
[43]  2.133657  1.765051  2.907751  2.223789  4.123978  3.066311  4.014976
[50]  4.375807  3.689392  2.647397  3.048005  2.426740  1.364755  2.744958
[57]  4.874926  3.503679  4.030208  3.069646
```

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

plot(x)
```



Now lets try out `kmeans()`

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

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

Cluster means:
          x         y
1 -3.057816  3.149715
2  3.149715 -3.057816

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] 61.96499 61.96499
 (between_SS / total_SS =  90.3 %)

Available components:
```

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

```r
attributes(km)
```

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

$class
[1] "kmeans"
```

> Q. How many points in each cluster

```r
km$size
```

```
[1] 30 30
```

> Q. What components of your result object details 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
```

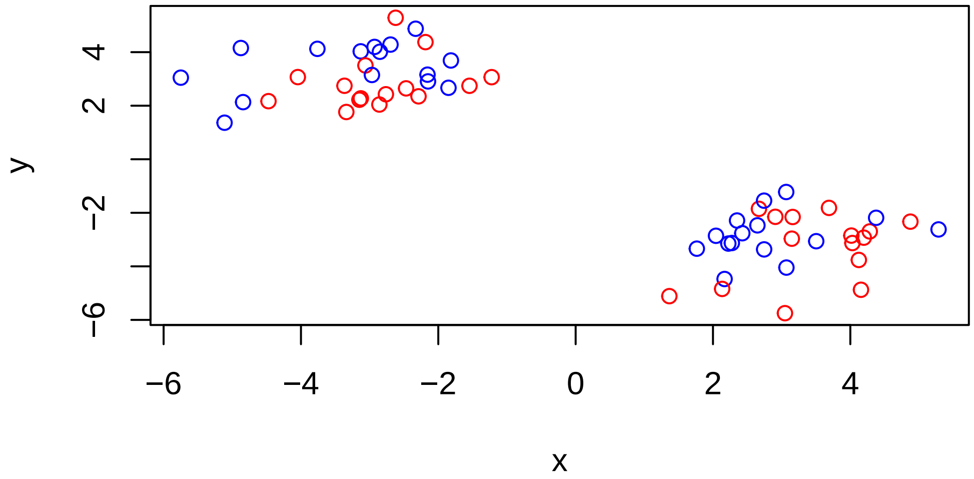
> Q. What are centers/mean calues of each cluster?

```r
km$centers
```

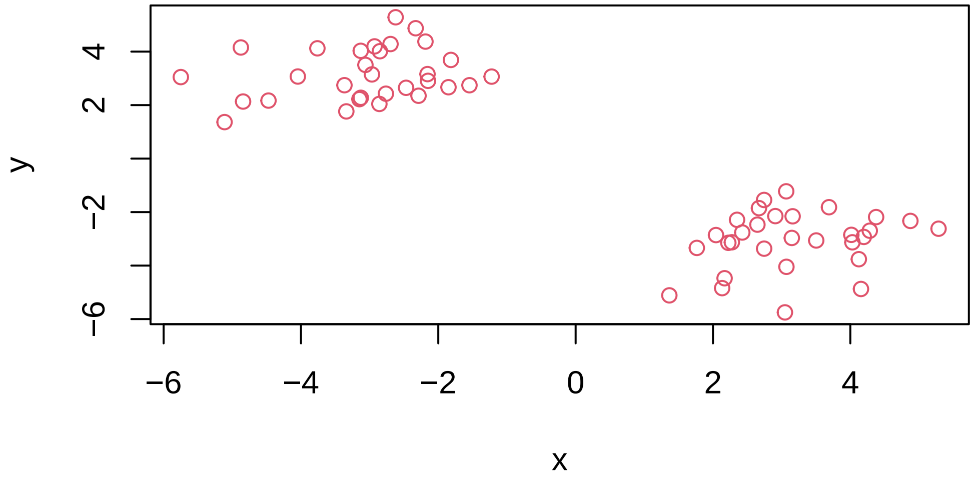
```
          x         y
1 -3.057816  3.149715
2  3.149715 -3.057816
```

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

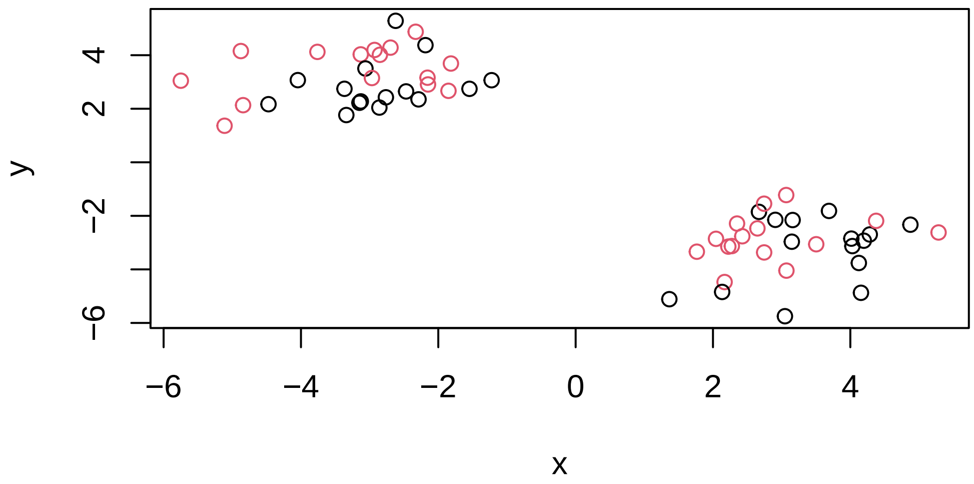
```r
plot(x, col=c("red", "blue"))
```



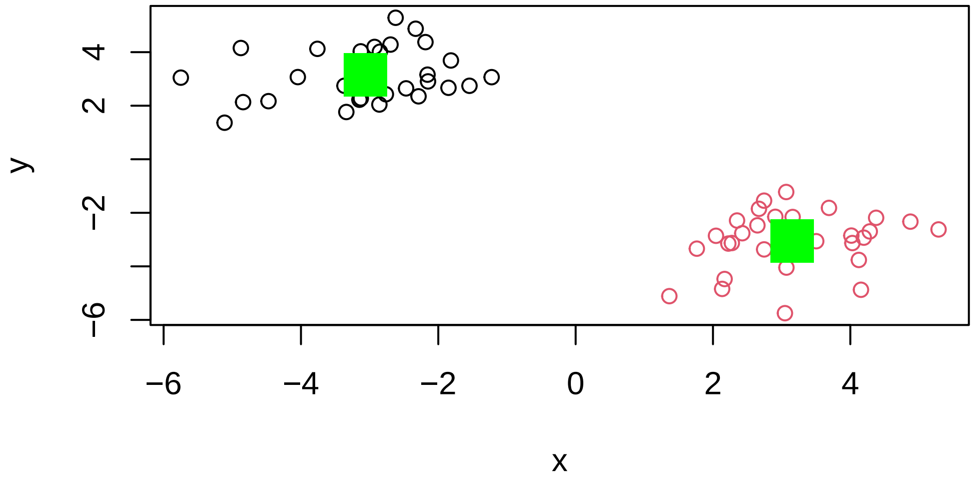
```
plot(x, col=2)
```



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



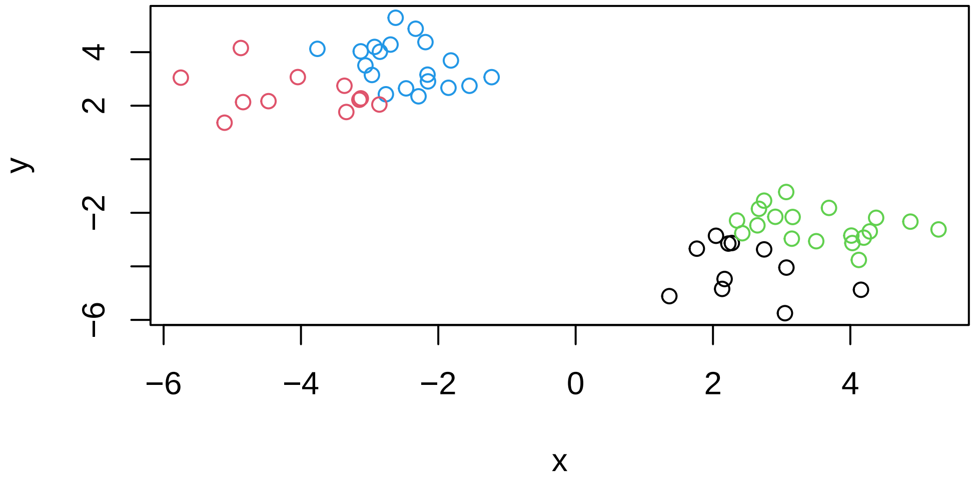
```
plot(x, col=km$cluster)
points(km$centers, col="green", pch=15, cex=3)
```



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

```r
km4 <- kmeans(x, centers=4)

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



###Hierarchical Clustering

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

THe main function in base R for this called `hclust()`. This function does not take our input data directly but wnats a "distance matrix" that details how (dis)similar all our input points are to each other.

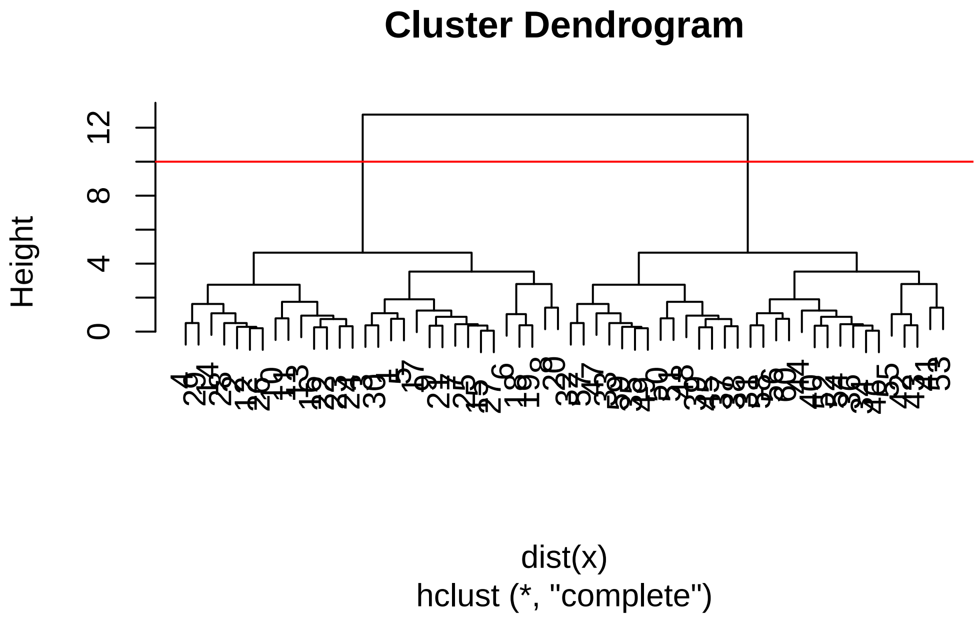
```
hc <- hclust( dist(x) )
hc
```

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

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

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

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

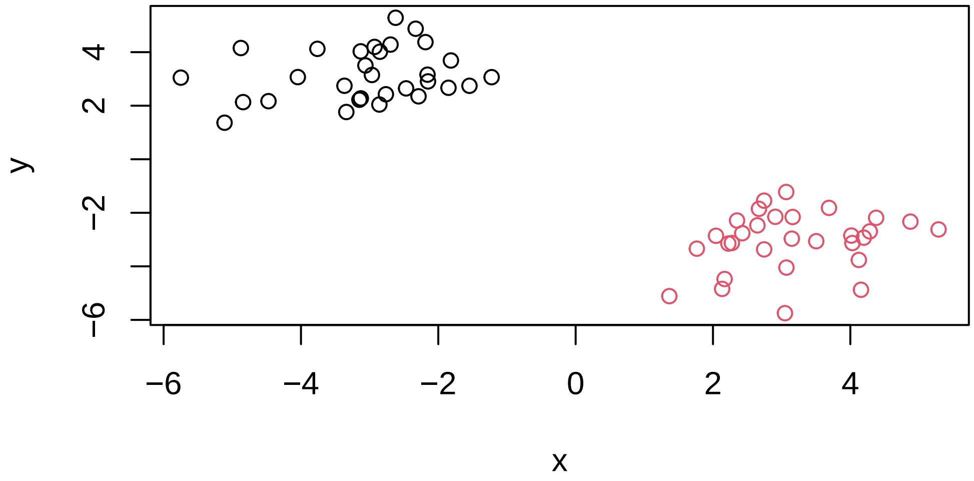


To get my main result (my cluster membership vector) I need to "cut" my tree using the function `cutree()`

```
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
```

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



principal component analysis (PCA)

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

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

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
x
```

```
                     X England Wales Scotland N.Ireland
1               Cheese     105   103      103        66
2        Carcass_meat      245   227      242       267
3          Other_meat      685   803      750       586
4                 Fish     147   160      122        93
5       Fats_and_oils      193   235      184       209
6               Sugars     156   175      147       139
7      Fresh_potatoes      720   874      566      1033
```

```
8           Fresh_Veg      253  265      171       143
9           Other_Veg      488  570      418       355
10 Processed_potatoes      198  203      220       187
11      Processed_Veg      360  365      337       334
12        Fresh_fruit     1102 1137      957       674
13            Cereals     1472 1582     1462      1494
14            Beverages     57   73       53        47
15        Soft_drinks     1374 1256     1572      1506
16   Alcoholic_drinks      375  475      458       135
17      Confectionery       54   64       62        41
```

```
ncol(x)
```

```
[1] 5
```

```
nrow(x)
```

```
[1] 17
```

```
rownames(x) <- x[,1]
x <- x[,-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
```

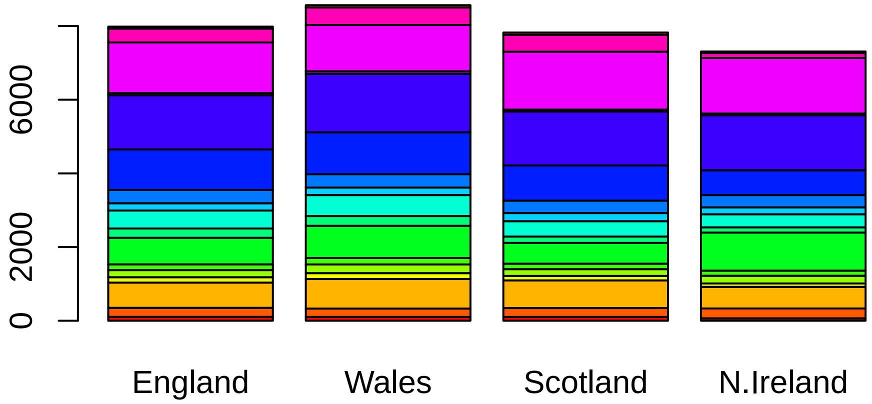
```
dim(x)
```

```
[1] 17  4
```

```
x <- read.csv(url, row.names=1)
head(x)
```

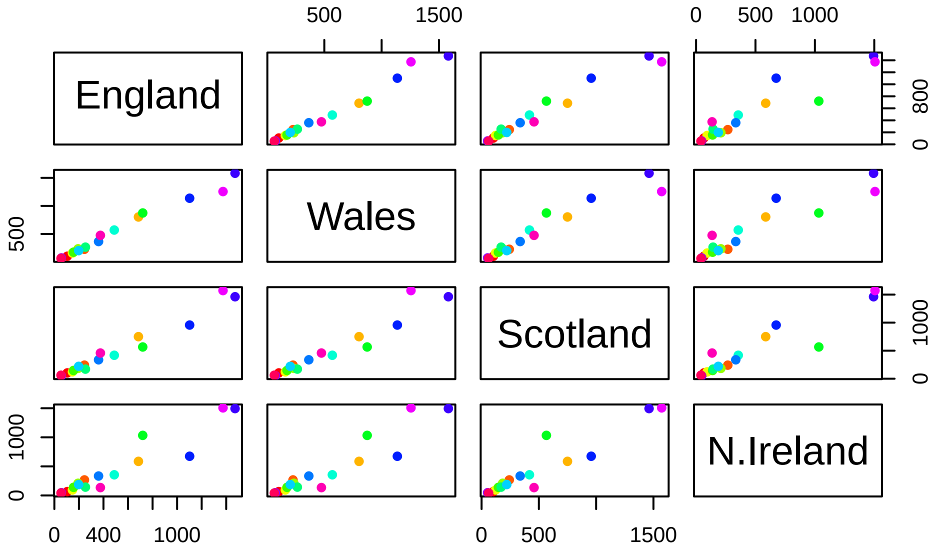
```
               England Wales Scotland N.Ireland
Cheese             105   103      103        66
Carcass_meat       245   227      242       267
Other_meat         685   803      750       586
Fish               147   160      122        93
Fats_and_oils      193   235      184       209
Sugars             156   175      147       139
```

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



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

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



so the paris plot is useful for small datasets but it can be lots of work to interpret and gets intractable for larger datasets.

So PCA to the rescue….

The main function to do PCA in base R is called `prcomp()`. This function wants the trasnpose of our data in this case.

```
#t(x)
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
```

```
attributes(pca)
```

```
$names
```

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

$class
[1] "prcomp"
```

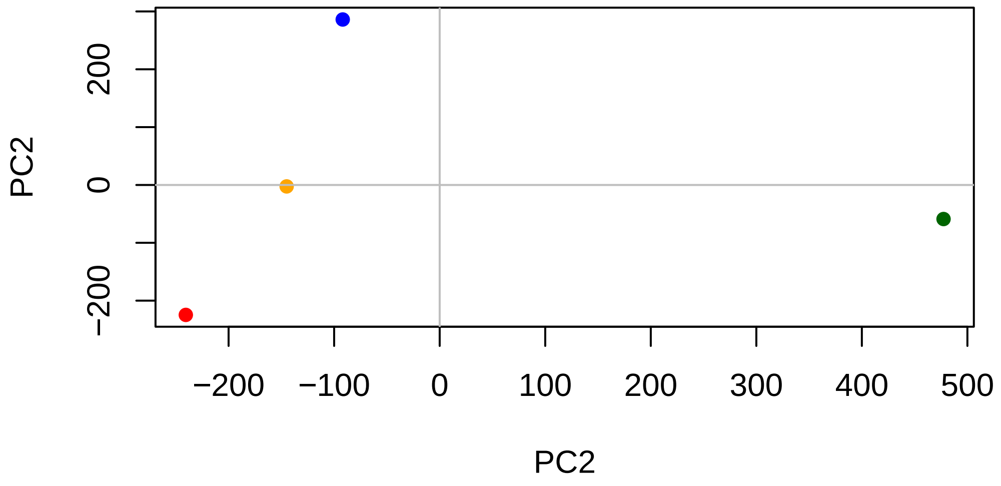
```
pca$x
```

```
                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
```

A major PCA result viz is called a "PCA plot" (a.k.a: a score plot, biplot, PC1 vs PC2 plot, ordienation plot)

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

abline(h=0, col="gray")
abline(v=0, col="gray")
```



Another important output from PCA is called the “loadings” vector or the “rotation component - this tells us how much the original variable (the food in this case) contributes to the new Pcs.

```
pca$rotation
```

```
                           PC1          PC2         PC3          PC4
Cheese             -0.056955380  0.016012850  0.02394295 -0.409382587
Carcass_meat        0.047927628  0.013915823  0.06367111  0.729481922
Other_meat         -0.258916658 -0.015331138 -0.55384854  0.331001134
Fish               -0.084414983 -0.050754947  0.03906481  0.022375878
Fats_and_oils      -0.005193623 -0.095388656 -0.12522257  0.034512161
Sugars             -0.037620983 -0.043021699 -0.03605745  0.024943337
Fresh_potatoes      0.401402060 -0.715017078 -0.20668248  0.021396007
Fresh_Veg          -0.151849942 -0.144900268  0.21382237  0.001606882
Other_Veg          -0.243593729 -0.225450923 -0.05332841  0.031153231
Processed_potatoes -0.026886233  0.042850761 -0.07364902 -0.017379680
Processed_Veg      -0.036488269 -0.045451802  0.05289191  0.021250980
Fresh_fruit        -0.632640898 -0.177740743  0.40012865  0.227657348
Cereals            -0.047702858 -0.212599678 -0.35884921  0.100043319
Beverages          -0.026187756 -0.030560542 -0.04135860 -0.018382072
Soft_drinks         0.232244140  0.555124311 -0.16942648  0.222319484
Alcoholic_drinks   -0.463968168  0.113536523 -0.49858320 -0.273126013
Confectionery      -0.029650201  0.005949921 -0.05232164  0.001890737
```

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