class07QD

Ilyas Darif A16577084

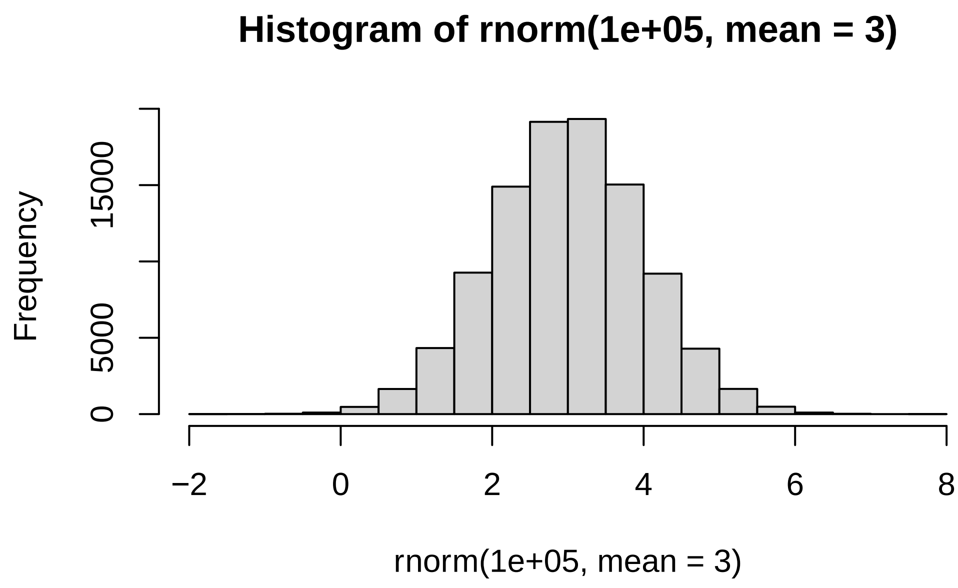
2024-04-23

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 dimensionallity reduction

Clustering

lets start with "k-means" cluttering the main function in base R for the is `means()`

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

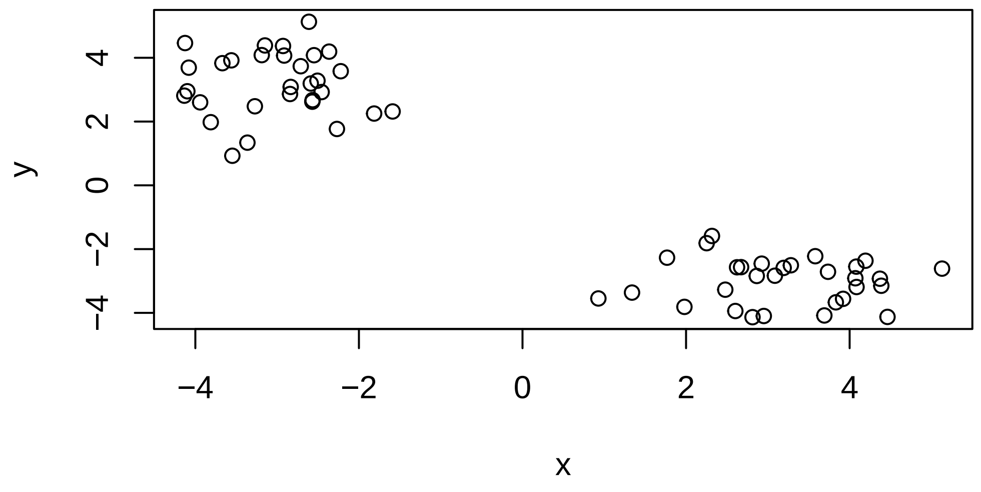


```r
tmp <- c(rnorm(30, -3), rnorm(30, +3))

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

```
               x          y
 [1,] -2.6113127  5.1299379
 [2,] -4.1261385  4.4627565
 [3,] -2.5490062  4.0816916
 [4,] -3.8114809  1.9800675
 [5,] -2.8344824  3.0857448
 [6,] -4.0974978  2.9505656
 [7,] -3.3632147  1.3392605
 [8,] -2.3642414  4.1934734
 [9,] -2.8421212  2.8643826
[10,] -2.5066374  3.2811309
[11,] -2.5657316  2.6727083
[12,] -2.2222934  3.5793545
[13,] -1.5879509  2.3166419
[14,] -2.7111841  3.7354913
[15,] -2.2686093  1.7676810
[16,] -3.9412907  2.6022936
[17,] -2.4557176  2.9250352
[18,] -4.1346654  2.8126393
[19,] -4.0804667  3.6907071
[20,] -3.5474158  0.9278268
[21,] -1.8138030  2.2516766
[22,] -3.1880890  4.0844581
[23,] -2.5697118  2.6233017
[24,] -3.2722725  2.4793893
[25,] -3.1490273  4.3870648
[26,] -2.9280196  4.3700703
[27,] -3.5594731  3.9197163
[28,] -2.9141401  4.0696375
[29,] -2.5886518  3.1934598
[30,] -3.6692415  3.8313418
[31,]  3.8313418 -3.6692415
[32,]  3.1934598 -2.5886518
[33,]  4.0696375 -2.9141401
[34,]  3.9197163 -3.5594731
[35,]  4.3700703 -2.9280196
[36,]  4.3870648 -3.1490273
```

```
[37,]  2.4793893 -3.2722725
[38,]  2.6233017 -2.5697118
[39,]  4.0844581 -3.1880890
[40,]  2.2516766 -1.8138030
[41,]  0.9278268 -3.5474158
[42,]  3.6907071 -4.0804667
[43,]  2.8126393 -4.1346654
[44,]  2.9250352 -2.4557176
[45,]  2.6022936 -3.9412907
[46,]  1.7676810 -2.2686093
[47,]  3.7354913 -2.7111841
[48,]  2.3166419 -1.5879509
[49,]  3.5793545 -2.2222934
[50,]  2.6727083 -2.5657316
[51,]  3.2811309 -2.5066374
[52,]  2.8643826 -2.8421212
[53,]  4.1934734 -2.3642414
[54,]  1.3392605 -3.3632147
[55,]  2.9505656 -4.0974978
[56,]  3.0857448 -2.8344824
[57,]  1.9800675 -3.8114809
[58,]  4.0816916 -2.5490062
[59,]  4.4627565 -4.1261385
[60,]  5.1299379 -2.6113127
```

```r
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  3.186984 -3.009130
2 -3.009130  3.186984

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 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] 43.05918 43.05918
 (between_SS / total_SS =  93.0 %)

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"
```

> Q1. how many points in each cluster?

```r
km$size
```

```
[1] 30 30
```

> Q2. what commponant of your result object details cluster assignment/membership

```r
km$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 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
```

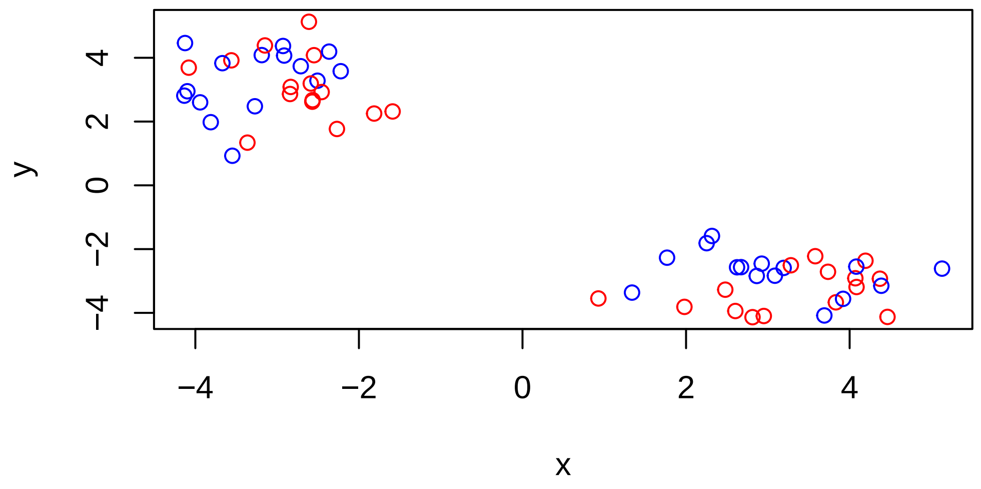
> Q3. what are centers/mean values of each cluster

```r
km$centers
```

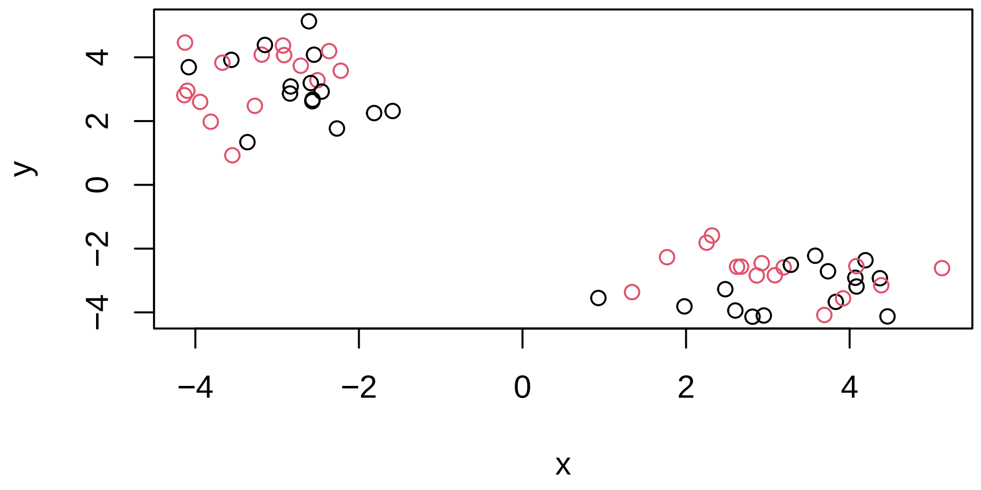
```
          x         y
1  3.186984 -3.009130
2 -3.009130  3.186984
```

> Q4. make a plot of your data showing your clustering results (groupings/clusters and cluster centers)

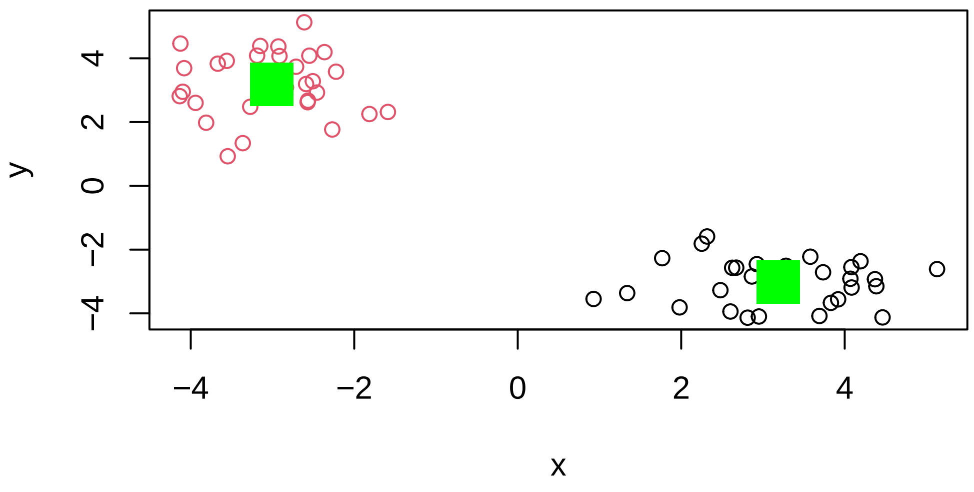
```r
plot(x, col=c("red", "blue"))
```



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

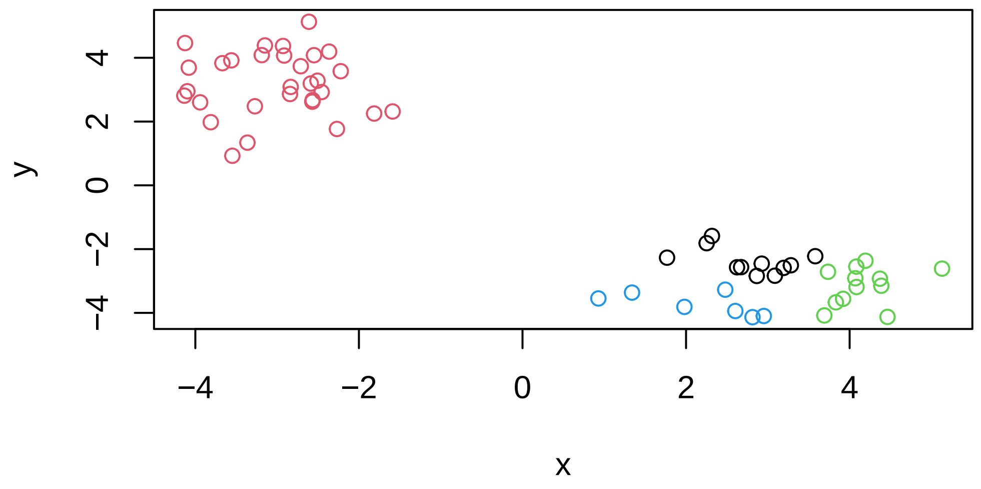


```r
plot(x, col=km$cluster)

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



Q5. run `kmeans()` again and cluster in 4 groups and plot the results.

```r
km4 <- kmeans(x, centers = 4)
plot(x, col=km4$cluster)
```



hierarchial clustering

this form of clustering aims to reveal the structure in your data by progessively grouping points into a ever smaller number of clusters

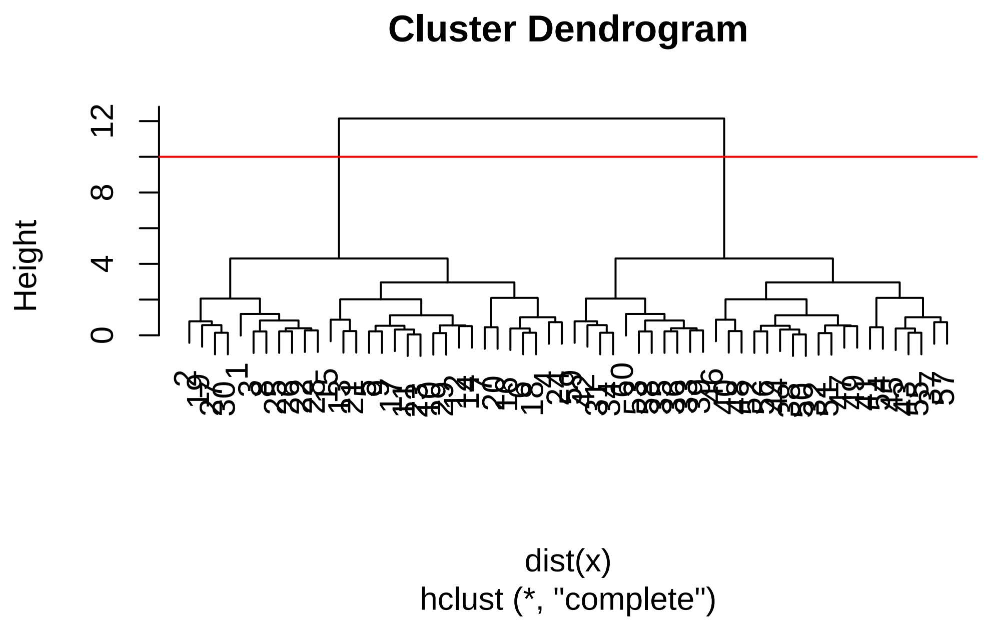
the maion function in base R for this called `chlust()` . this function does not take our input data directly but wants a "distance matrix" that details how (dis)similar our inout points are to each other

```
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 useful (unlike that from kmeans) but there is a useful `plot()` method

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

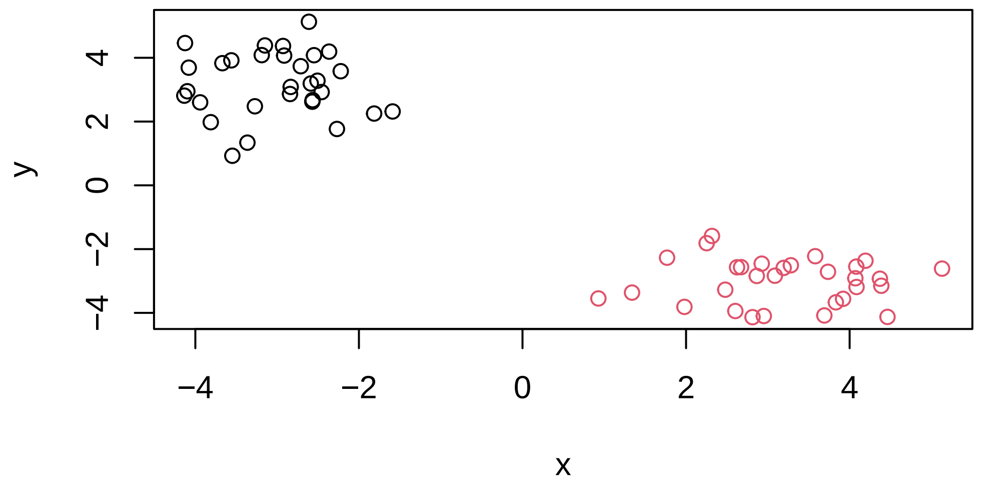


to get my results (my cluster membership vector) i need to "cut" my tree using the function `cutree()`

```
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 2 2
```

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



Principal Component Analysis (PSA)

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 ect

Q1.

```
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
```

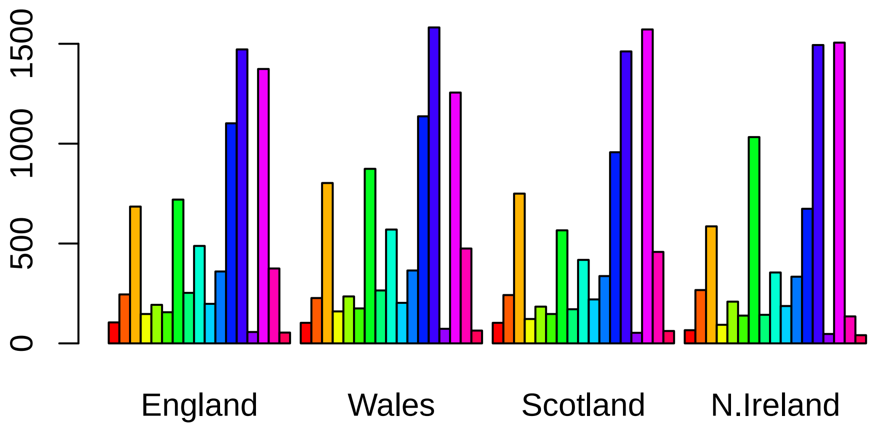
```
dim(x)
```

```
[1] 17  4
```

Q2. the `row.names = 1` way because it was a little more simple to use

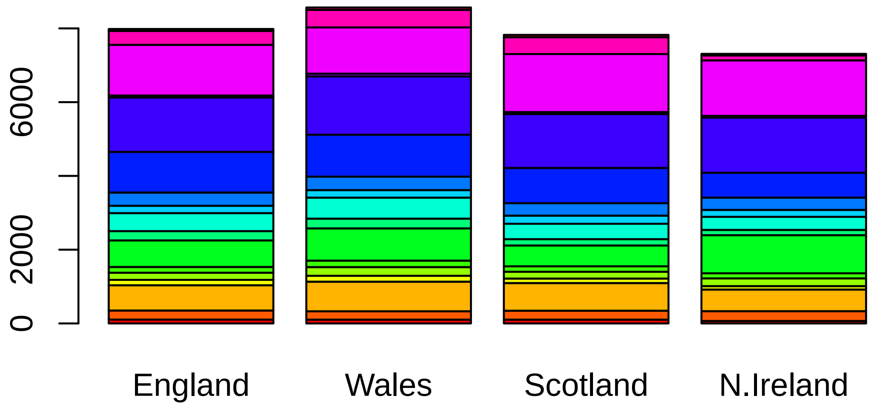
Q3.

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



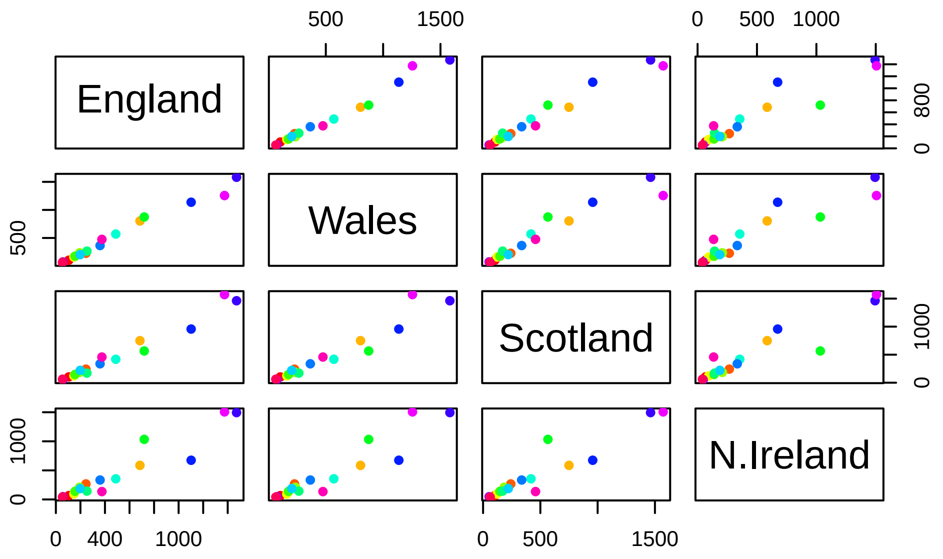
to make the plot the other bar style you change `beside=T` to `beside=F`

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



Q4(5). it means these are the axis for each plot

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



so the paris plot is usful for small datasets but it can be lots of work to interpret and gets interactable for longer datasets

So PCA to the rescue... the main function to do PCA in base Ris called `prcomp()`. this function wants the transpof our datain this case.

```
t(x)
```

```
          Cheese Carcass_meat  Other_meat  Fish Fats_and_oils  Sugars
England      105          245         685   147           193     156
Wales        103          227         803   160           235     175
Scotland     103          242         750   122           184     147
N.Ireland     66          267         586    93           209     139
          Fresh_potatoes  Fresh_Veg  Other_Veg  Processed_potatoes
England              720        253        488                 198
Wales                874        265        570                 203
Scotland             566        171        418                 220
N.Ireland           1033        143        355                 187
          Processed_Veg  Fresh_fruit  Cereals  Beverages Soft_drinks
England             360         1102     1472         57        1374
Wales               365         1137     1582         73        1256
Scotland            337          957     1462         53        1572
```

```
N.Ireland            334          674     1494        47          1506
          Alcoholic_drinks  Confectionery
England                375             54
Wales                  475             64
Scotland               458             62
N.Ireland              135             41
```

```r
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
```

```r
attributes(pca)
```

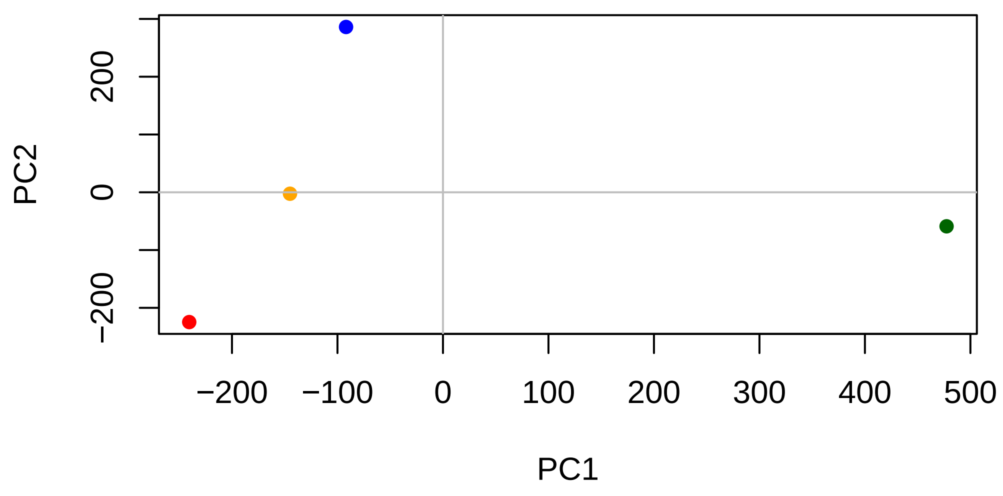
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$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
```

A MAJOR pcA result viz is called a “PCA plot” (aka a score plot, biplot, pc1 vs pc2 plot, ordination plot)

```r
mycols <- c("orange", "red", "blue", "darkgreen")
plot(pca$x[,1], pca$x[,2], col=mycols, pch=16, xlab="PC1", 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 variabls (the foods in this case)

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