Class07: Matchine Learning 1

Yang Liu

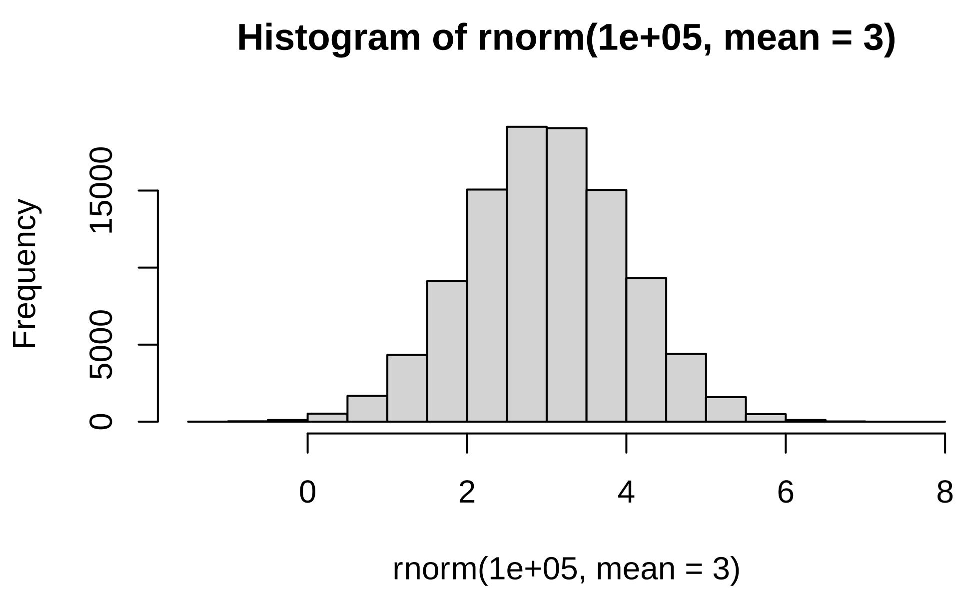
Today we will start out multi-part exploration of some key machine learning methods. We will begin with clustering--finding groupings in data, and then dimensionality reduction.

##Clustering

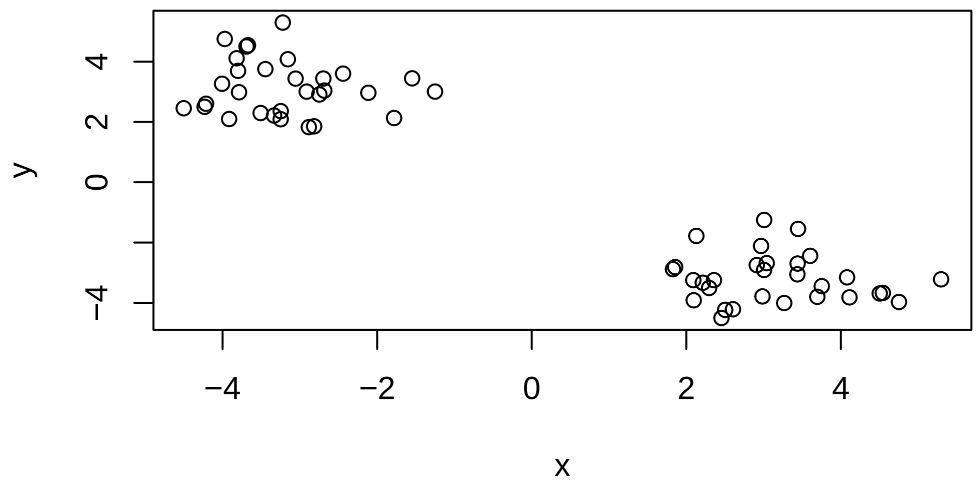
Let's start with "k-means" clustering

The main function in base R for this 'kmeans()'.

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



```r
tmp <- c(rnorm(30, -3), rnorm(30, +3))
x <- cbind(x=tmp, y=rev(tmp))
plot(x)
```



Now let's try out 'kmeans()'

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

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

Cluster means:
          x         y
1  3.142578 -3.190014
2 -3.190014  3.142578

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] 42.823 42.823
 (between_SS / total_SS =  93.4 %)

Available components:

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

```
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?

```
km$size
```

```
[1] 30 30
```

Q. What component of your result object details cluster assignment/membership?

```
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
```

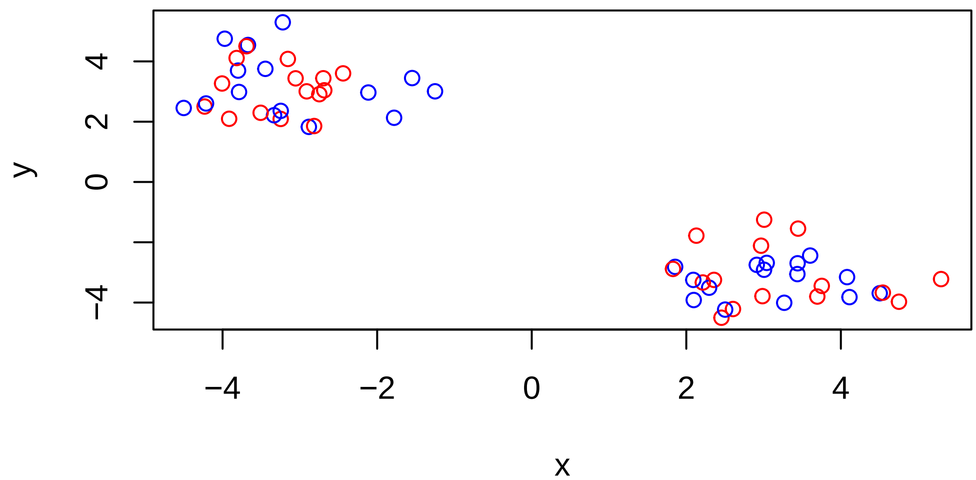
Q. what are centers/mean values of each cluster?

```
km$centers
```

```
          x         y
1  3.142578 -3.190014
2 -3.190014  3.142578
```

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

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

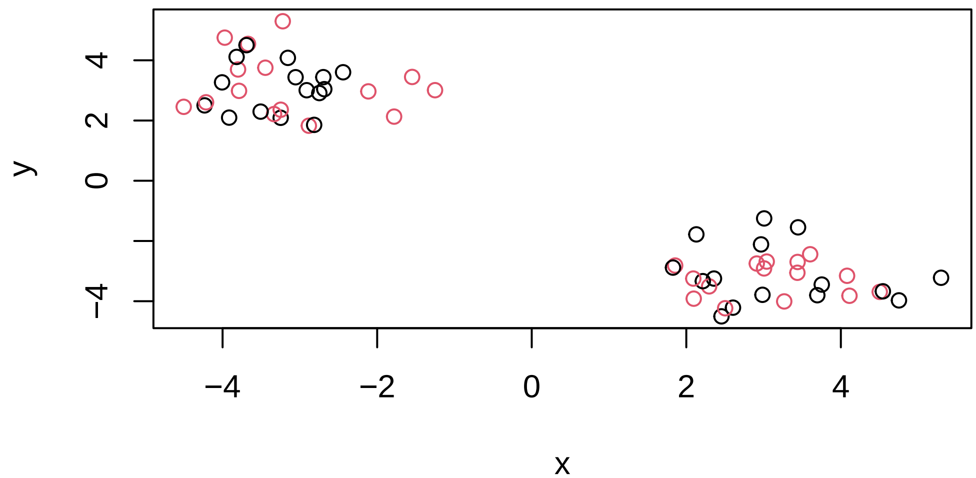



```r
c(1:5) + c(100, 1)
```

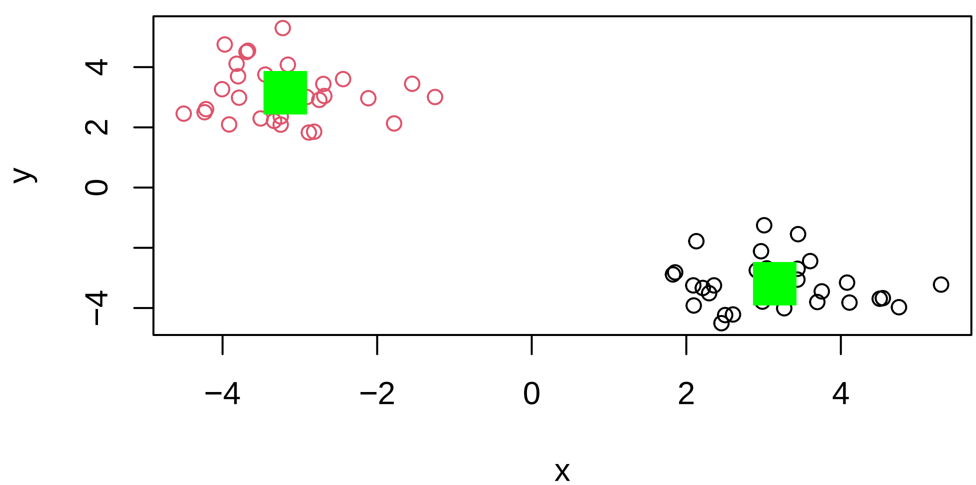
```
Warning in c(1:5) + c(100, 1): longer object length is not a multiple of
shorter object length
```

```
[1] 101   3 103   5 105
```

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




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

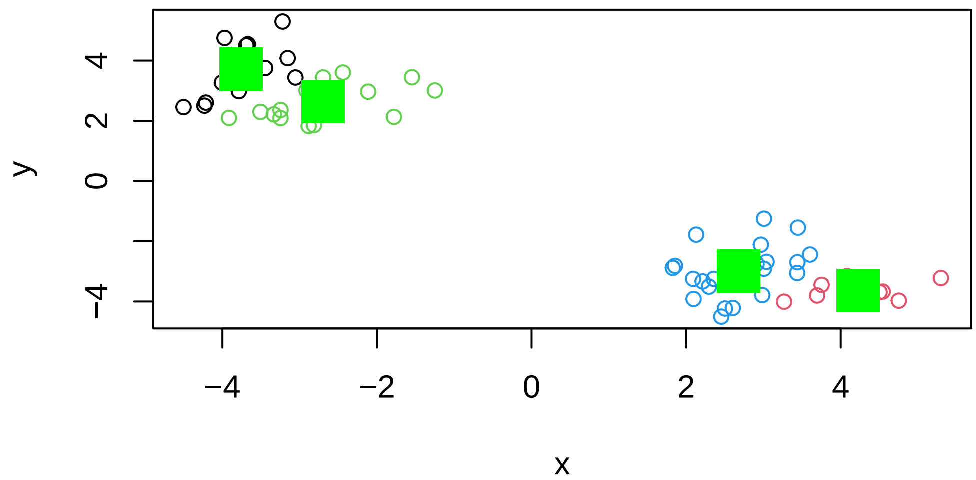


Q. Run 'kmeans' again and cluster in 4 groups and plot the results.

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

```
K-means clustering with 4 clusters of sizes 14, 9, 16, 21

Cluster means:
          x         y
1 -3.755358  3.713319
2  4.222433 -3.642528
3 -2.695338  2.643180
4  2.679783 -2.996080

Clustering vector:
 [1] 1 1 1 3 3 3 3 3 3 3 3 1 1 1 3 1 1 1 3 1 3 3 1 3 3 1 3 3 1 1 2 2 4 4 4 4 4 2
[39] 4 4 2 4 4 2 2 4 2 2 2 4 4 4 4 4 4 4 4 4 4 4

Within cluster sum of squares by cluster:
[1] 12.634305  3.812884 13.248033 21.384833
 (between_SS / total_SS =  96.0 %)

Available components:

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

```r
plot(x, col=km4$cluster)
points(km4$centers, col="green", pch=15, cex=3)
```



##Hierarchical Clustering

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

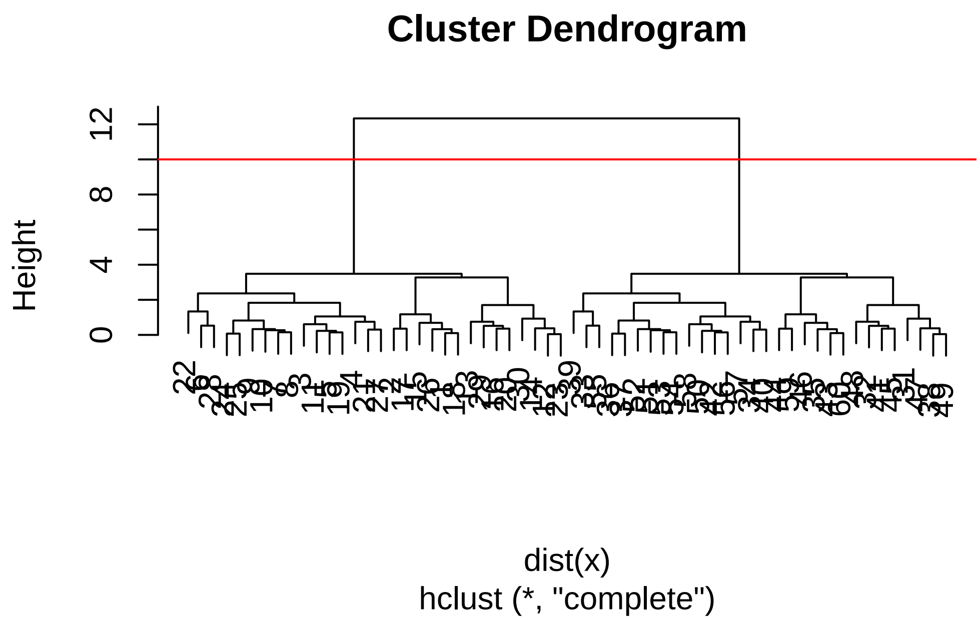
The main fucntion in base R for this called 'hclust()' This function does not take out input data directiely but wants a "distance matrix" that details how (dis)similar all our inout points are to each otehr.

```
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()' mehtod.

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

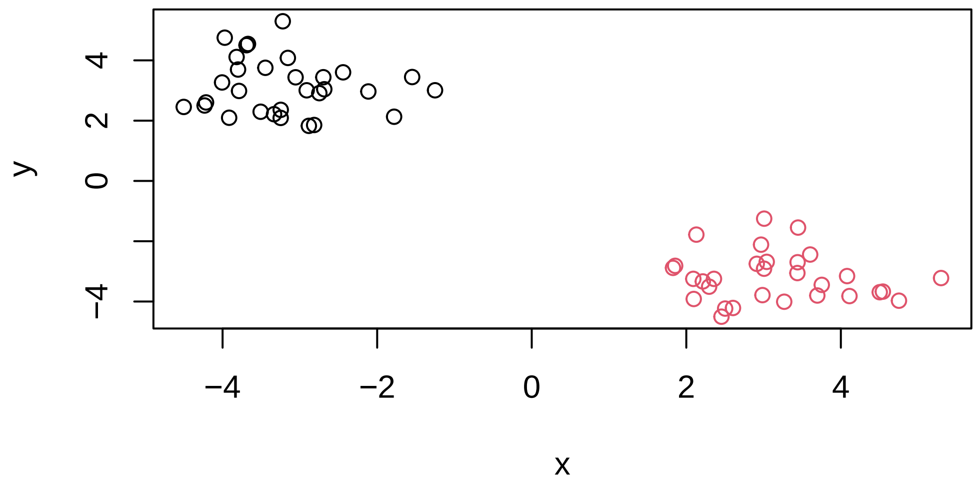


To get my main result (My cluster membership vector) I need to "cut" my tree using the function 'cutree()'

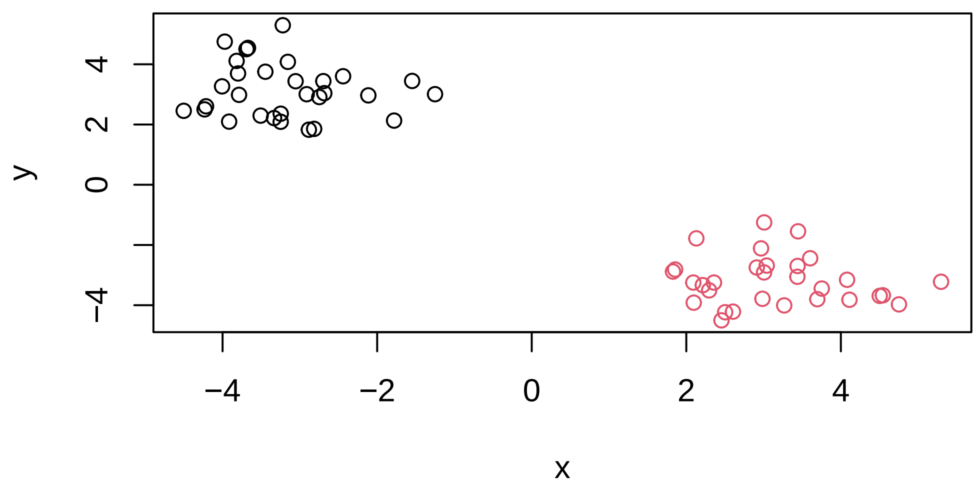
```r
grps <- cutree(hc, h=10)
grps
```

```
 [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
```

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




```
plot(x, col=cutree(hc, h=6))
```



#Principal Component Analysis (PCA)

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

Data import

Read data about crazy eacting trends in the UK and N. Ireland.

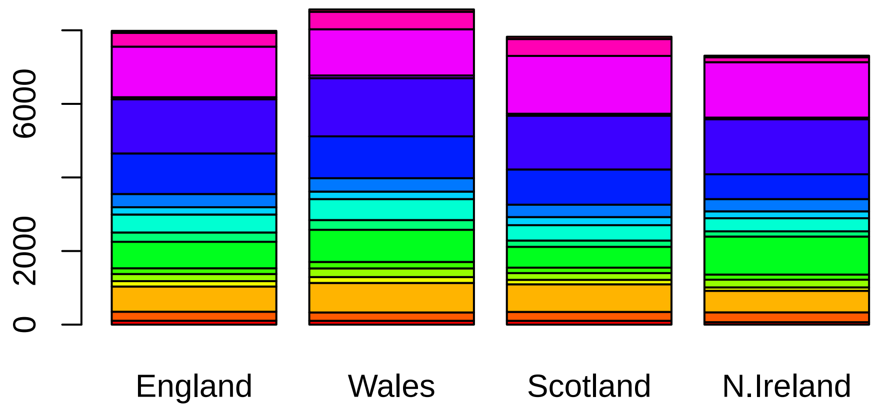
```r
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names = 1)
x
```

```
                    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
```

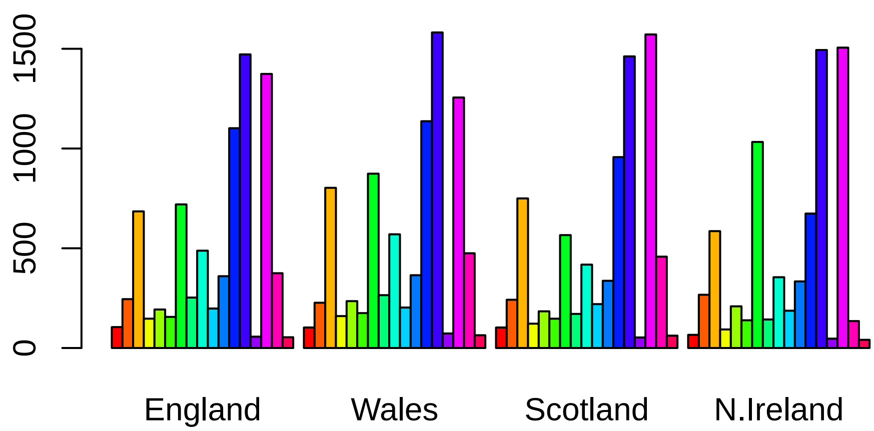
```r
dim(x)
```

```
[1] 17  4
```

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

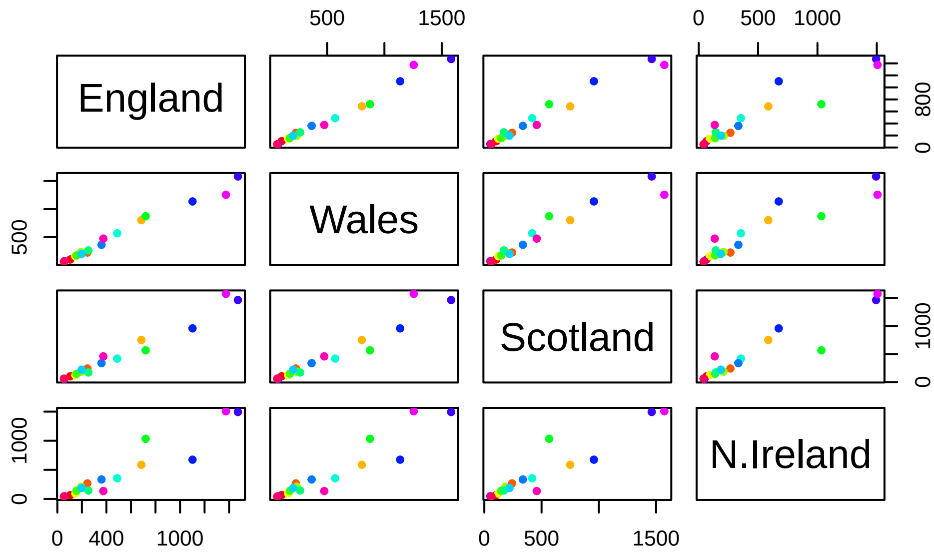



```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



The so-called "pairs" plot can be useful for small datasets:

```
#rainbow(nrow(x))
pairs(x, col=rainbow(nrow(x)), pch=20)
```



So the paris plot is useful for small datasets but it can be lots of work to interpret and get untractable for larger datasets.

So PCA to the rescue...

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

```
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)
```

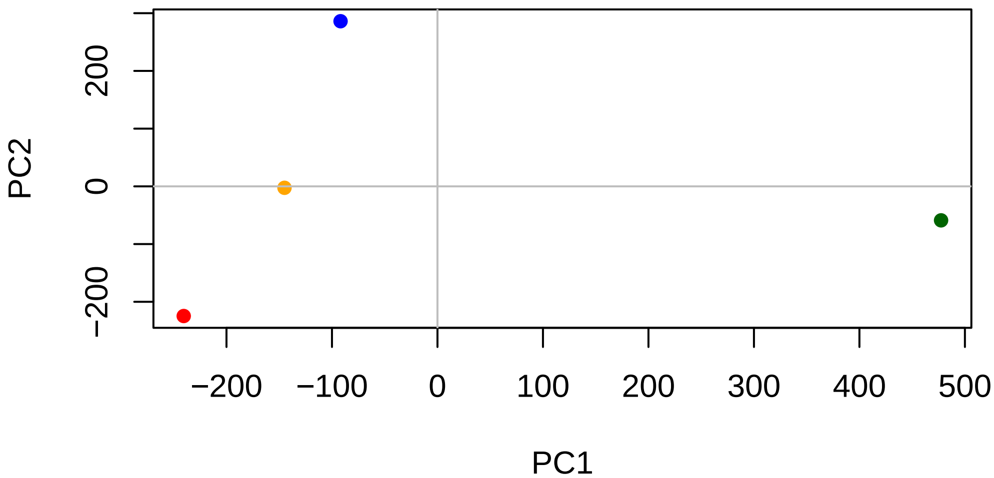
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$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" (a.k.a. a score plot, biplot, PC1 vs. PC2 plot, ordienation 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 tell us how much the original variables (the food in this case) contributes to the new PCs

```
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.

PCA of RNA-seq data

```
url2 <- "https://tinyurl.com/expression-CSV"
rna.data <- read.csv(url2, row.names=1)
head(rna.data)
```

```
       wt1 wt2  wt3  wt4 wt5 ko1 ko2 ko3 ko4 ko5
gene1  439 458  408  429 420  90  88  86  90  93
gene2  219 200  204  210 187 427 423 434 433 426
gene3 1006 989 1030 1017 973 252 237 238 226 210
gene4  783 792  829  856 760 849 856 835 885 894
gene5  181 249  204  244 225 277 305 272 270 279
gene6  460 502  491  491 493 612 594 577 618 638
```