Class07: Machine Learning 1

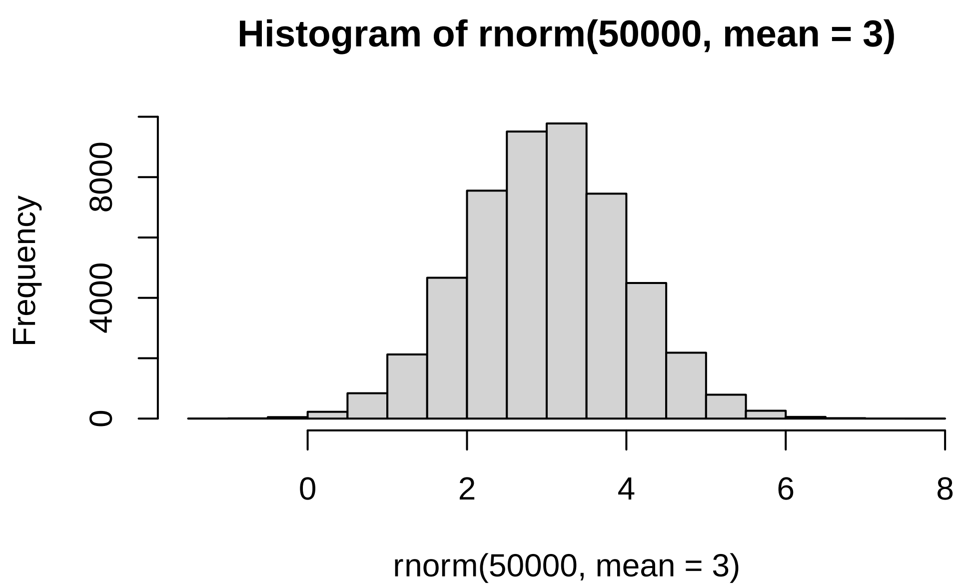
Xiaoxuan Teng

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.

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



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

```r
tmp <- c(rnorm(30, mean = 3), rnorm(30, -3))
head(tmp,5)
```

```
[1] 3.043486 3.099859 3.752496 2.570857 3.574098
```

```r
tmprev <- rev(tmp)
head(tmprev, 5)
```

```
[1] -4.298645 -3.412140 -2.736373 -3.626369 -2.182122
```

```r
cbind(tmp, tmprev)
```

```
           tmp    tmprev
 [1,]  3.043486 -4.298645
 [2,]  3.099859 -3.412140
 [3,]  3.752496 -2.736373
 [4,]  2.570857 -3.626369
 [5,]  3.574098 -2.182122
 [6,]  1.998864 -1.090543
 [7,]  4.002716 -3.448597
 [8,]  2.373313 -3.939813
 [9,]  4.902900 -2.178152
[10,]  3.095312 -2.658011
[11,]  1.321822 -3.430090
[12,]  2.482382 -3.244236
[13,]  4.929141 -4.436760
[14,]  3.226788 -2.556968
[15,]  2.340204 -2.841512
[16,]  1.442351 -2.756285
[17,]  3.742581 -4.651425
[18,]  2.394229 -1.228944
[19,]  2.919445 -3.971646
[20,]  1.863077 -3.812187
[21,]  5.102882 -4.945030
[22,]  3.035461 -2.600502
[23,]  3.951028 -1.333797
[24,]  3.719334 -1.073864
```

```
[25,]  2.188846 -3.241831
[26,]  3.585748 -2.776552
[27,]  2.012962 -3.003520
[28,]  3.356575 -3.280432
[29,]  3.026929 -3.373480
[30,]  3.318531 -4.359435
[31,] -4.359435  3.318531
[32,] -3.373480  3.026929
[33,] -3.280432  3.356575
[34,] -3.003520  2.012962
[35,] -2.776552  3.585748
[36,] -3.241831  2.188846
[37,] -1.073864  3.719334
[38,] -1.333797  3.951028
[39,] -2.600502  3.035461
[40,] -4.945030  5.102882
[41,] -3.812187  1.863077
[42,] -3.971646  2.919445
[43,] -1.228944  2.394229
[44,] -4.651425  3.742581
[45,] -2.756285  1.442351
[46,] -2.841512  2.340204
[47,] -2.556968  3.226788
[48,] -4.436760  4.929141
[49,] -3.244236  2.482382
[50,] -3.430090  1.321822
[51,] -2.658011  3.095312
[52,] -2.178152  4.902900
[53,] -3.939813  2.373313
[54,] -3.448597  4.002716
[55,] -1.090543  1.998864
[56,] -2.182122  3.574098
[57,] -3.626369  2.570857
[58,] -2.736373  3.752496
[59,] -3.412140  3.099859
[60,] -4.298645  3.043486
```

```r
# reverse elements `rev()`
rev(1:5)
```

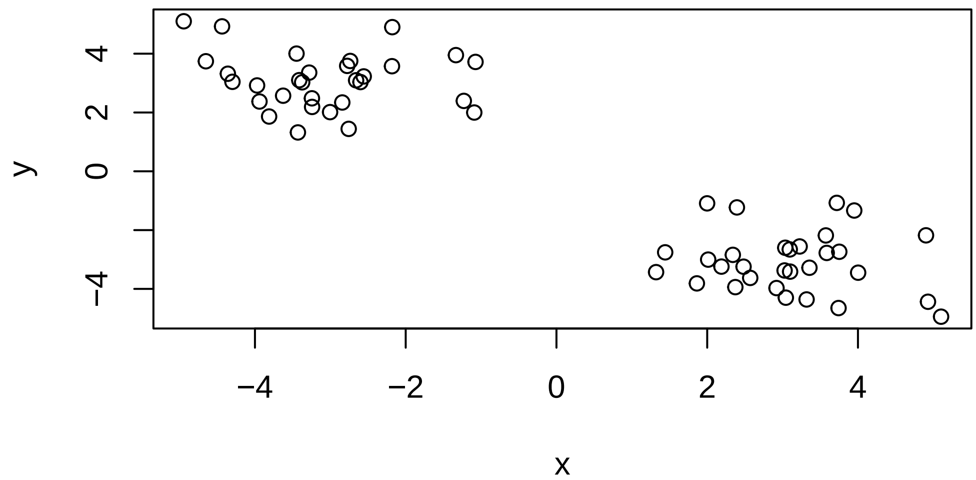
```
[1] 5 4 3 2 1
```

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

```
              x         y
 [1,]  3.043486 -4.298645
 [2,]  3.099859 -3.412140
 [3,]  3.752496 -2.736373
 [4,]  2.570857 -3.626369
 [5,]  3.574098 -2.182122
 [6,]  1.998864 -1.090543
 [7,]  4.002716 -3.448597
 [8,]  2.373313 -3.939813
 [9,]  4.902900 -2.178152
[10,]  3.095312 -2.658011
[11,]  1.321822 -3.430090
[12,]  2.482382 -3.244236
[13,]  4.929141 -4.436760
[14,]  3.226788 -2.556968
[15,]  2.340204 -2.841512
[16,]  1.442351 -2.756285
[17,]  3.742581 -4.651425
[18,]  2.394229 -1.228944
[19,]  2.919445 -3.971646
[20,]  1.863077 -3.812187
[21,]  5.102882 -4.945030
[22,]  3.035461 -2.600502
[23,]  3.951028 -1.333797
[24,]  3.719334 -1.073864
[25,]  2.188846 -3.241831
[26,]  3.585748 -2.776552
[27,]  2.012962 -3.003520
[28,]  3.356575 -3.280432
[29,]  3.026929 -3.373480
[30,]  3.318531 -4.359435
[31,] -4.359435  3.318531
[32,] -3.373480  3.026929
[33,] -3.280432  3.356575
[34,] -3.003520  2.012962
[35,] -2.776552  3.585748
[36,] -3.241831  2.188846
[37,] -1.073864  3.719334
[38,] -1.333797  3.951028
```

```
[39,] -2.600502  3.035461
[40,] -4.945030  5.102882
[41,] -3.812187  1.863077
[42,] -3.971646  2.919445
[43,] -1.228944  2.394229
[44,] -4.651425  3.742581
[45,] -2.756285  1.442351
[46,] -2.841512  2.340204
[47,] -2.556968  3.226788
[48,] -4.436760  4.929141
[49,] -3.244236  2.482382
[50,] -3.430090  1.321822
[51,] -2.658011  3.095312
[52,] -2.178152  4.902900
[53,] -3.939813  2.373313
[54,] -3.448597  4.002716
[55,] -1.090543  1.998864
[56,] -2.182122  3.574098
[57,] -3.626369  2.570857
[58,] -2.736373  3.752496
[59,] -3.412140  3.099859
[60,] -4.298645  3.043486
```

```
plot(x)
```



Run `kmeans()` asking for two clusters:

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

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

Cluster means:
          x         y
1  3.079141 -3.082975
2 -3.082975  3.079141

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] 57.88508 57.88508
 (between_SS / total_SS =  90.8 %)

Available components:
```

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

What is in this result object?

```
attributes(k)
```

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

$class
[1] "kmeans"
```

What are the cluster centers?

```
k$centers
```

```
          x         y
1  3.079141 -3.082975
2 -3.082975  3.079141
```

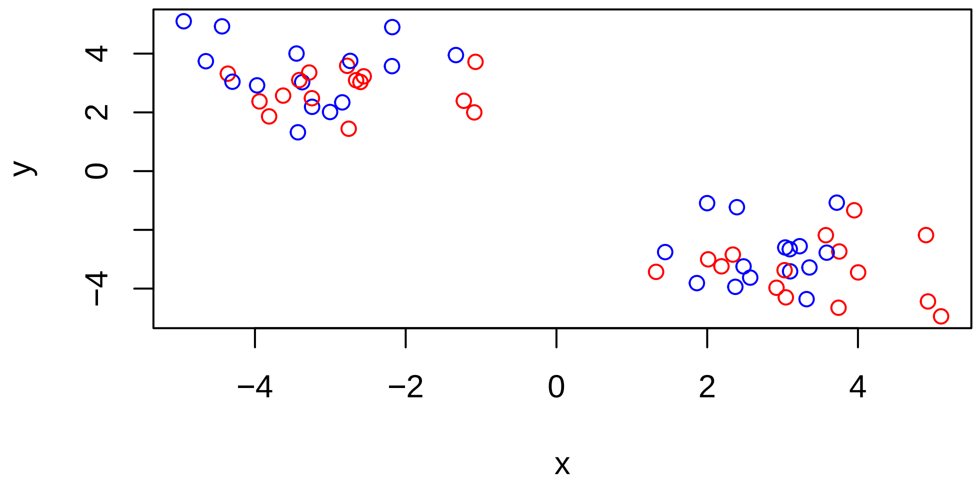
What is my clustering result? i.e. what cluster does each point reside in?

```
k$cluster
```

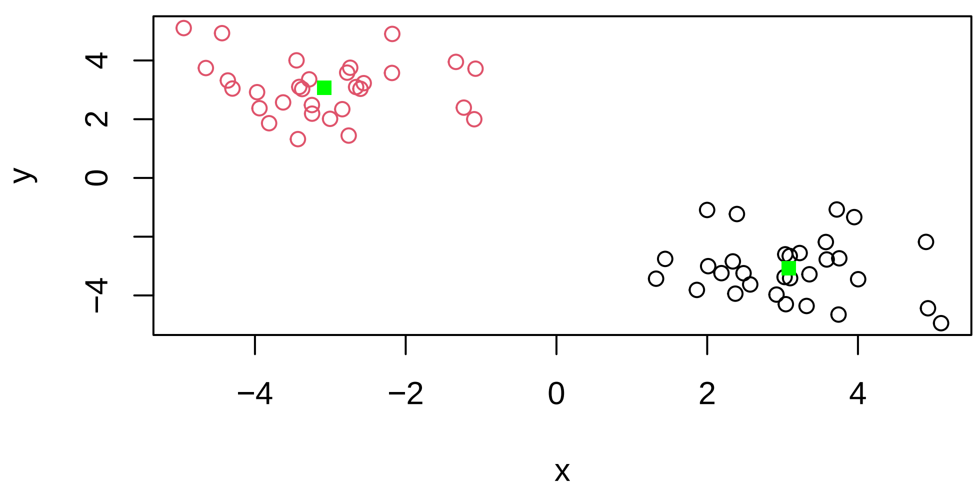
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

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

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




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

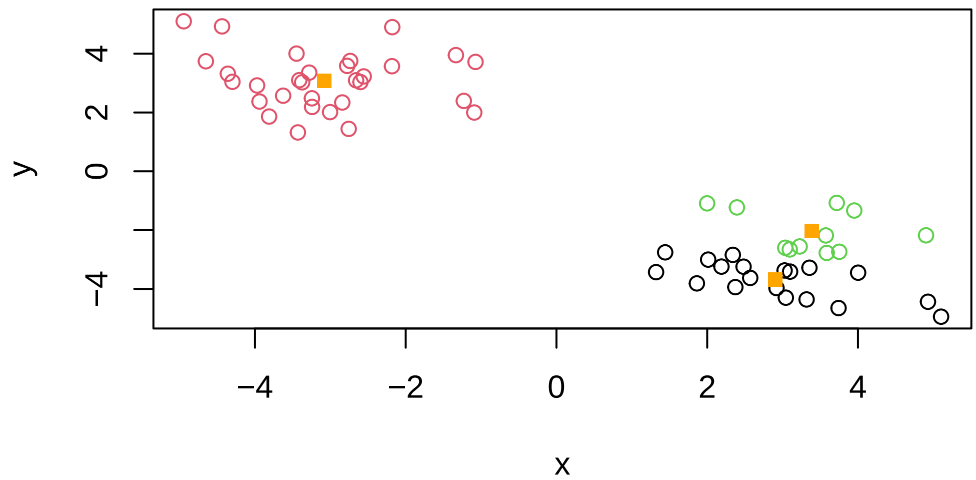


Q. Run kmeans and cluster into 3 grps and plot the result?

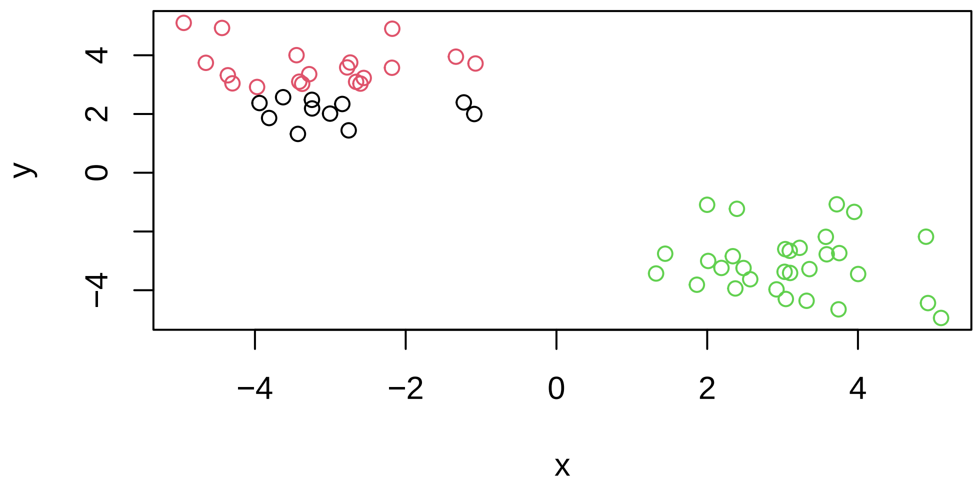
```r
k2 <- kmeans(x, centers=3, nstart=20)
k2
```

```
K-means clustering with 3 clusters of sizes 19, 30, 11

Cluster means:
          x         y
1  2.901998 -3.688075
2 -3.082975  3.079141
3  3.385114 -2.037803

Clustering vector:
 [1] 1 1 3 1 3 3 1 1 3 3 1 1 1 3 1 1 1 3 1 1 1 3 3 3 1 3 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] 26.18314 57.88508 11.10288
 (between_SS / total_SS =  92.4 %)

Available components:

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

```r
plot(x, col=k2$cluster)
points(k2$centers, pch = 15, col = "orange")
```




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



```
k$tot.withinss
```

```
[1] 115.7702
```

```
k3$tot.withinss
```

```
[1] 98.35993
```

The big limitation of `kmeans` is that it imposes a structure on your data (i.e. a clustering) that you ask for in the first place

Hierarchical Clustering

The main function in "base" R for this is called `hclust()`. It wants a distance matrix as input not the data itself.

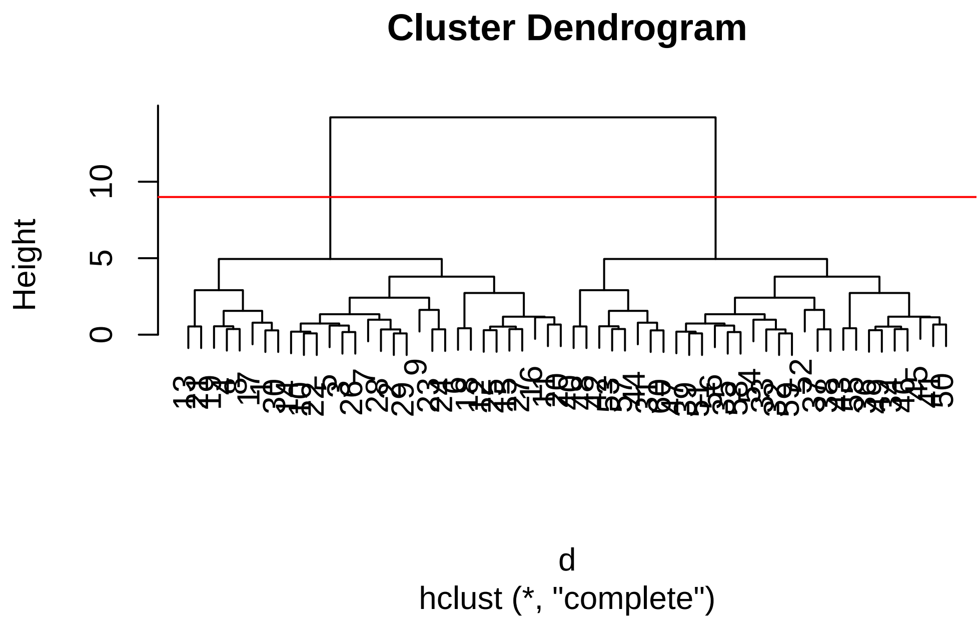
We can calculate a distance matrix in lots of different ways but here we will use the `dist()` function.

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

```

Call:
hclust(d = d)

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

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

```
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()`.

```r
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
```

```r
cutree(hc, k=4)
```

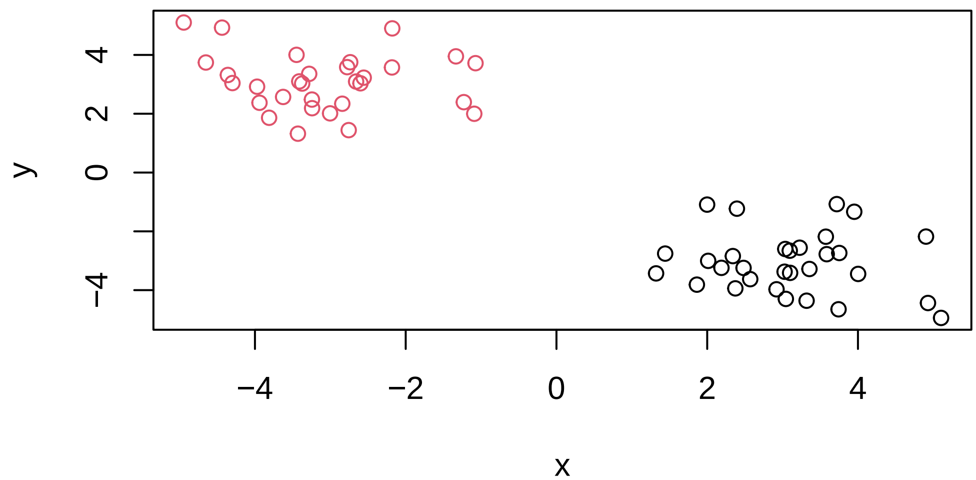
```
 [1] 1 2 2 1 2 2 2 1 2 2 2 2 1 2 2 2 1 2 1 2 1 2 2 2 2 2 2 2 2 1 3 4 4 4 4 4 4 4
[39] 4 3 4 3 4 3 4 4 4 3 4 4 4 4 3 4 4 4 3 4 4 3
```

```r
grps <- cutree(hc, k=2)
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. Plot our data(`x`) colored by our hclust result.

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



Principal Component Analysis (PCA)

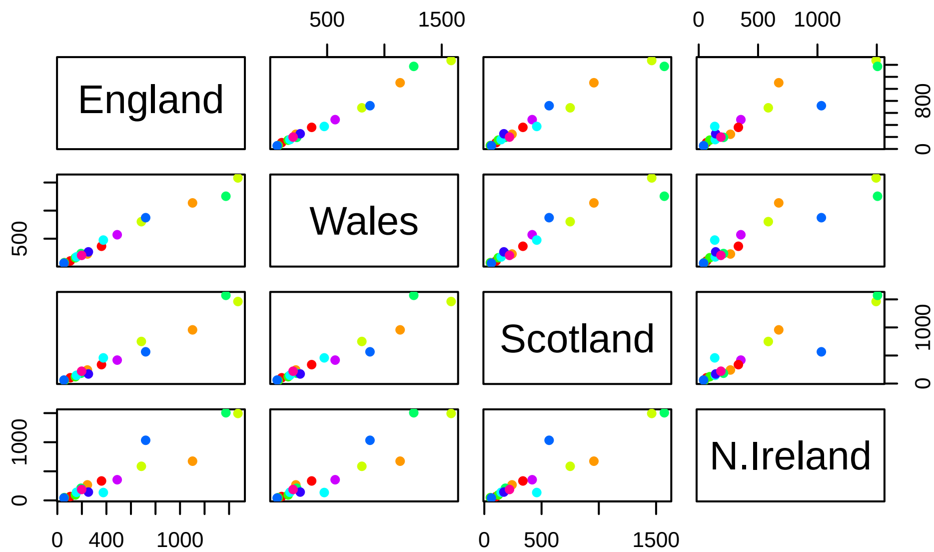
We will start with PCA of a tiny tiny dataset and make fun of stuff Barry eats.

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

```
[1] 17  4
```

One useful plot in this case (because we only have 4 contries to look accross) is a `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 to transpose of the data we have.

```
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"
```

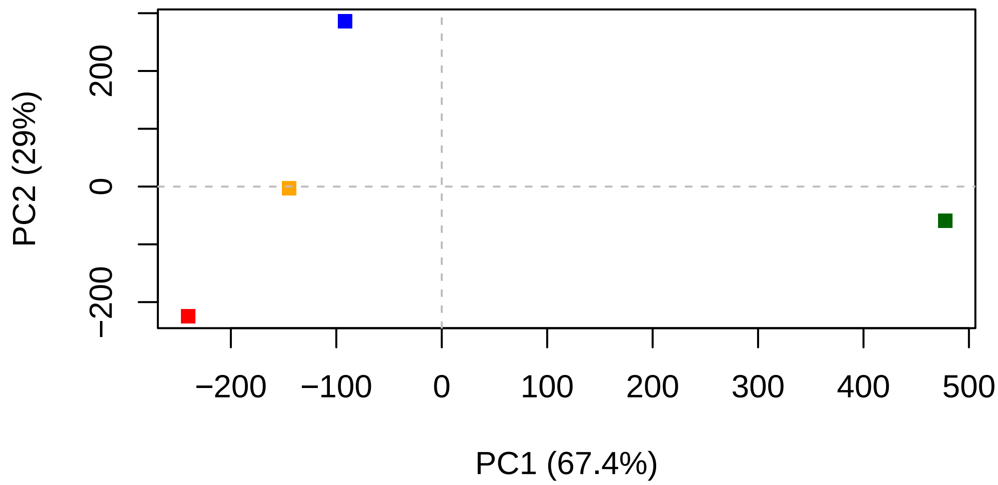
```r
pca$x
```

```
                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
```

```r
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
```

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


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