class 7: Machine Learning 1

Aigerim (PID: 09919142)

Exploring some machine learning methods. Namely clustering and dimensionality reduction approches.

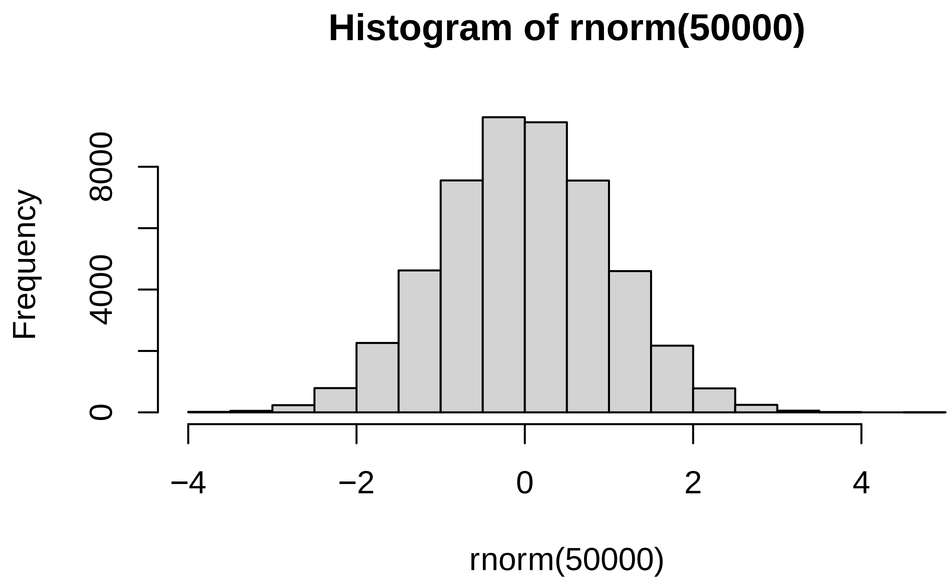
Kmeans clustering

The main function for k - means in "base" R is called `kmeans()`. Let's make first up some data to see kmen=ans works and to get at the results.

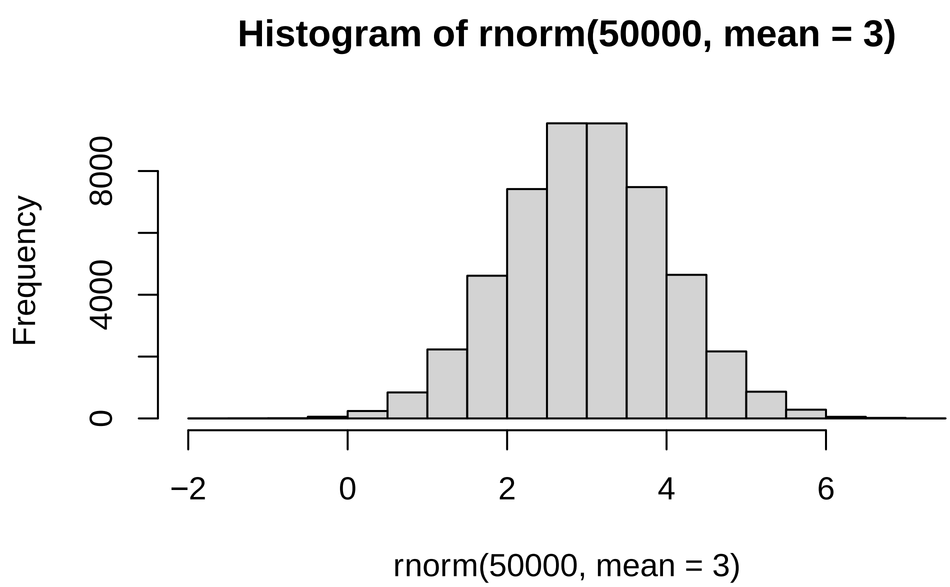
```r
rnorm(5)
```

```
[1]  0.06487613 -0.33287954  0.28107318  2.31714028  1.04781729
```

```r
hist(rnorm(50000))
```




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



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

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

```
 [1]  2.4389034  5.0142610  2.8171709  2.2782889  3.0967782  3.1754067
 [7]  3.6809926  4.0934686  3.7052081  2.7796220  3.7775896  2.6690034
[13]  3.4414440  3.0895565  2.1003376  2.7335203  1.9329317  5.0793661
[19]  4.6594650  4.7272341  4.3614329  3.4584010  3.6597683  2.5519351
[25]  3.3556437  4.8756042  4.5271217  3.4945413  3.0998483  3.6846583
[31] -3.1080375 -3.9030195 -1.7047774 -2.6890843 -4.2110303 -2.8903155
[37] -3.5943730 -2.6906120 -2.8438759 -2.2037142 -2.7526759  0.4504148
[43] -3.3990206 -3.2374743 -4.7566559 -3.2606406 -2.0955192 -3.2944333
[49] -4.2735382 -3.7910053 -3.2371721 -1.8277608 -2.6775428 -1.4250161
[55] -5.5585386 -2.5462781 -4.1108795 -3.6210604 -3.9083816 -3.2705247
```

```
rev(1:5)
```

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

```
rev(tmp)
```

```
 [1] -3.2705247 -3.9083816 -3.6210604 -4.1108795 -2.5462781 -5.5585386
 [7] -1.4250161 -2.6775428 -1.8277608 -3.2371721 -3.7910053 -4.2735382
[13] -3.2944333 -2.0955192 -3.2606406 -4.7566559 -3.2374743 -3.3990206
[19]  0.4504148 -2.7526759 -2.2037142 -2.8438759 -2.6906120 -3.5943730
[25] -2.8903155 -4.2110303 -2.6890843 -1.7047774 -3.9030195 -3.1080375
[31]  3.6846583  3.0998483  3.4945413  4.5271217  4.8756042  3.3556437
[37]  2.5519351  3.6597683  3.4584010  4.3614329  4.7272341  4.6594650
[43]  5.0793661  1.9329317  2.7335203  2.1003376  3.0895565  3.4414440
[49]  2.6690034  3.7775896  2.7796220  3.7052081  4.0934686  3.6809926
[55]  3.1754067  3.0967782  2.2782889  2.8171709  5.0142610  2.4389034
```

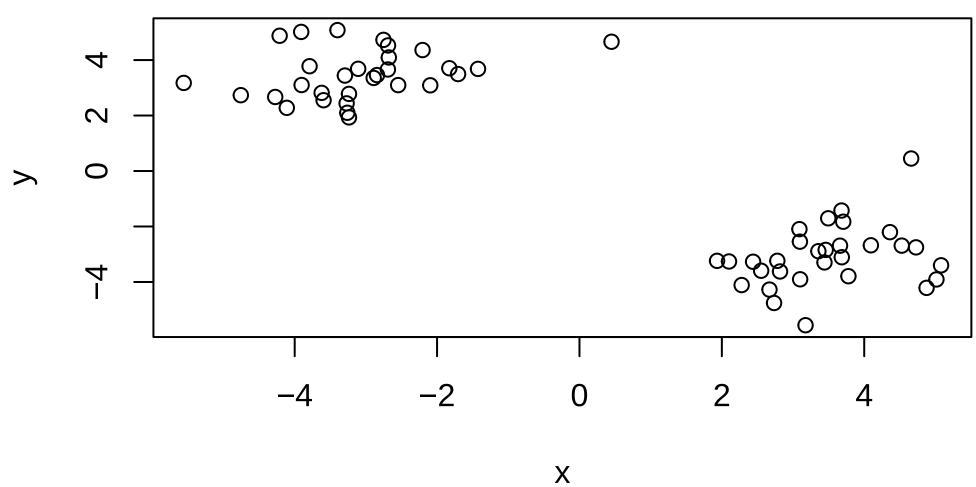
```
x <- cbind( x=tmp, y=rev(tmp))
x
```

```
               x          y
 [1,]  2.4389034 -3.2705247
```

```
 [2,]  5.0142610 -3.9083816
 [3,]  2.8171709 -3.6210604
 [4,]  2.2782889 -4.1108795
 [5,]  3.0967782 -2.5462781
 [6,]  3.1754067 -5.5585386
 [7,]  3.6809926 -1.4250161
 [8,]  4.0934686 -2.6775428
 [9,]  3.7052081 -1.8277608
[10,]  2.7796220 -3.2371721
[11,]  3.7775896 -3.7910053
[12,]  2.6690034 -4.2735382
[13,]  3.4414440 -3.2944333
[14,]  3.0895565 -2.0955192
[15,]  2.1003376 -3.2606406
[16,]  2.7335203 -4.7566559
[17,]  1.9329317 -3.2374743
[18,]  5.0793661 -3.3990206
[19,]  4.6594650  0.4504148
[20,]  4.7272341 -2.7526759
[21,]  4.3614329 -2.2037142
[22,]  3.4584010 -2.8438759
[23,]  3.6597683 -2.6906120
[24,]  2.5519351 -3.5943730
[25,]  3.3556437 -2.8903155
[26,]  4.8756042 -4.2110303
[27,]  4.5271217 -2.6890843
[28,]  3.4945413 -1.7047774
[29,]  3.0998483 -3.9030195
[30,]  3.6846583 -3.1080375
[31,] -3.1080375  3.6846583
[32,] -3.9030195  3.0998483
[33,] -1.7047774  3.4945413
[34,] -2.6890843  4.5271217
[35,] -4.2110303  4.8756042
[36,] -2.8903155  3.3556437
[37,] -3.5943730  2.5519351
[38,] -2.6906120  3.6597683
[39,] -2.8438759  3.4584010
[40,] -2.2037142  4.3614329
[41,] -2.7526759  4.7272341
[42,]  0.4504148  4.6594650
[43,] -3.3990206  5.0793661
[44,] -3.2374743  1.9329317
```

```
[45,] -4.7566559  2.7335203
[46,] -3.2606406  2.1003376
[47,] -2.0955192  3.0895565
[48,] -3.2944333  3.4414440
[49,] -4.2735382  2.6690034
[50,] -3.7910053  3.7775896
[51,] -3.2371721  2.7796220
[52,] -1.8277608  3.7052081
[53,] -2.6775428  4.0934686
[54,] -1.4250161  3.6809926
[55,] -5.5585386  3.1754067
[56,] -2.5462781  3.0967782
[57,] -4.1108795  2.2782889
[58,] -3.6210604  2.8171709
[59,] -3.9083816  5.0142610
[60,] -3.2705247  2.4389034
```

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



Run `kmeans()` asking for two clusters:

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

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

Cluster means:
          x         y
1 -3.081085  3.478650
2  3.478650 -3.081085

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] 59.45869 59.45869
 (between_SS / total_SS =  91.6 %)

Available components:

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

What is in this result object?

```r
attributes(k)
```

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

$class
[1] "kmeans"
```

What is cluster center?

```r
k$centers
```

```
          x         y
1 -3.081085  3.478650
2  3.478650 -3.081085
```

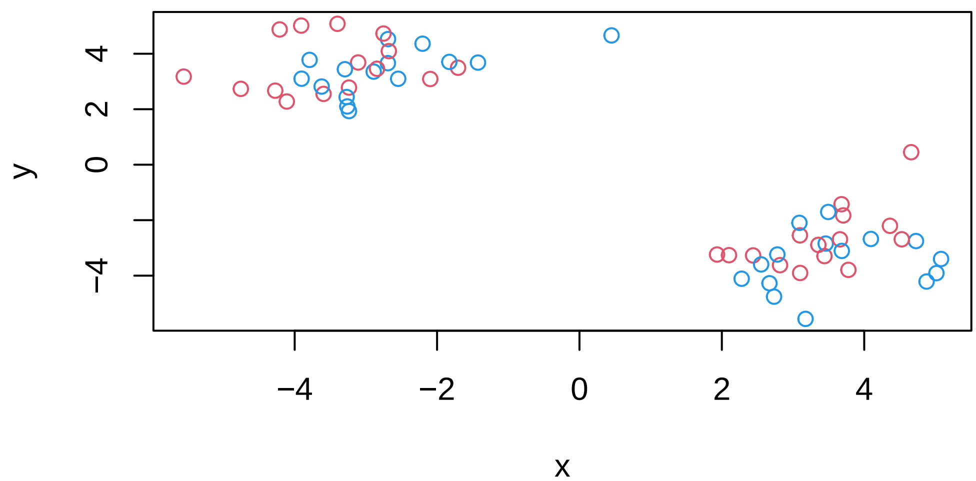
What is my clustering results? I.E. what cluster does each point rside in?

```r
k$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
```

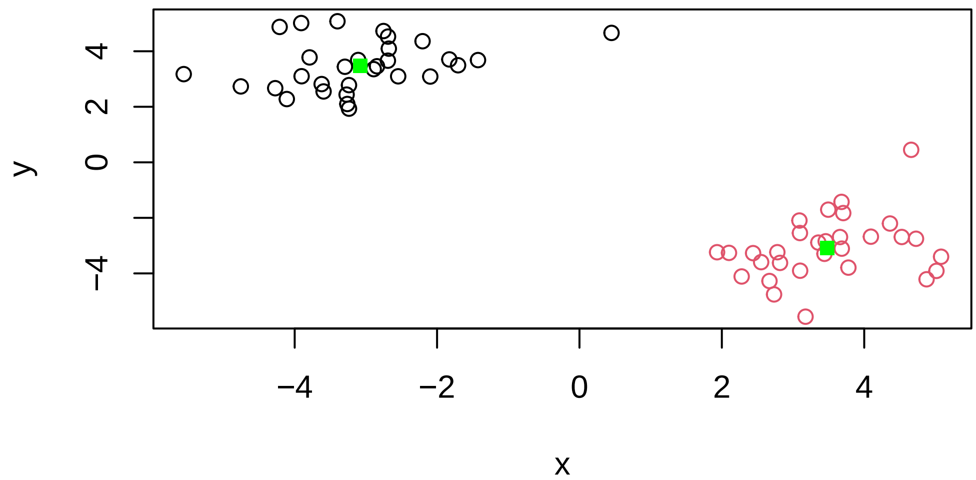
> Q. Plot your data `x` showing your clustering result and the center point for each cluster?

```r
plot(x, col = c(2,4))
```



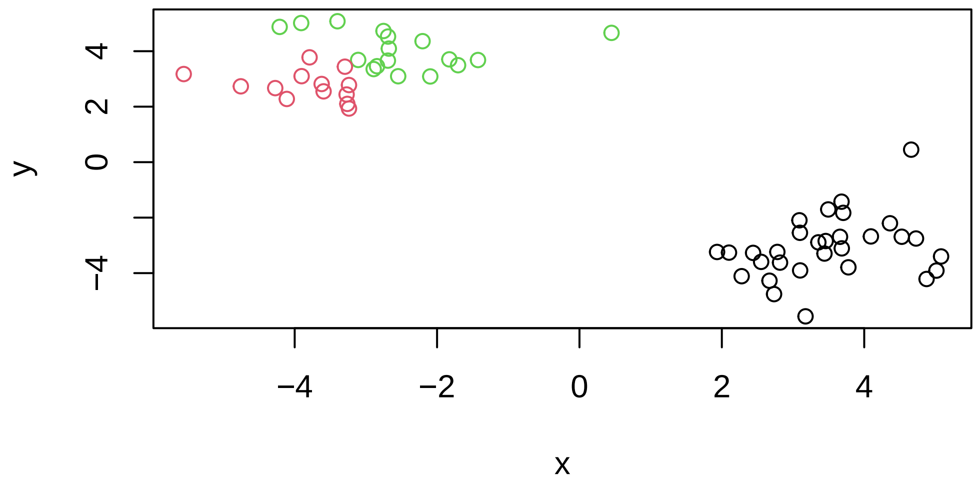
Center is shown as green dot

```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?

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




```
k$tot.withinss
```

```
[1] 118.9174
```

```
k3$tot.withinss
```

```
[1] 93.6708
```

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

#Hierarchical Clustering

The main function in "base" R for this is called `hlcust()`. 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.

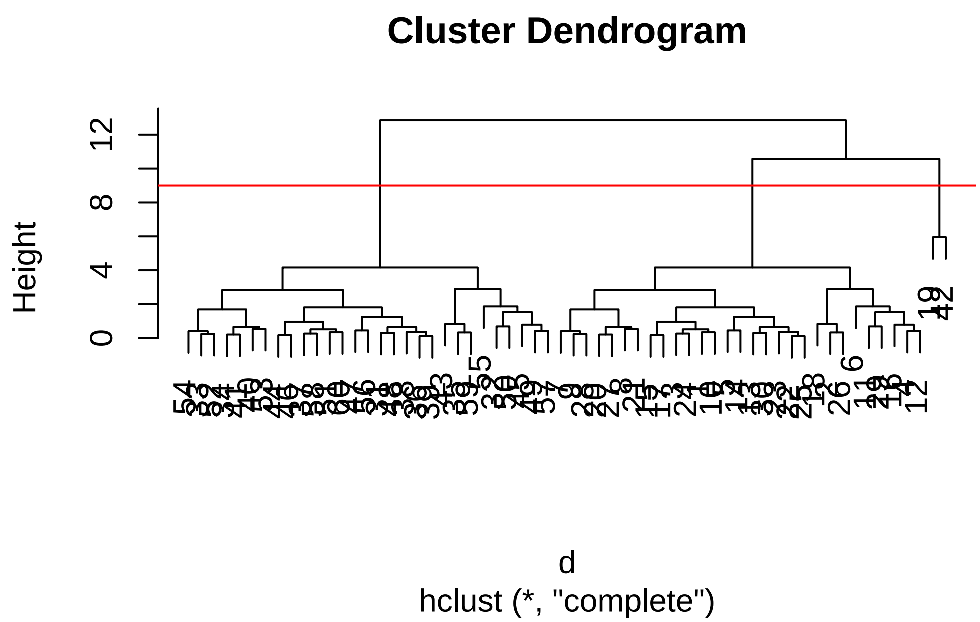
```
d <- dist(x)
hc <- hclust(d)
hc
```

```
Call:
hclust(d = d)

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

There is a specific plot method for hclust

```r
plot(hc)
abline(h=9, col="red")
```



To get the cluster membership vector we read 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 2 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3
[39] 3 3 3 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

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

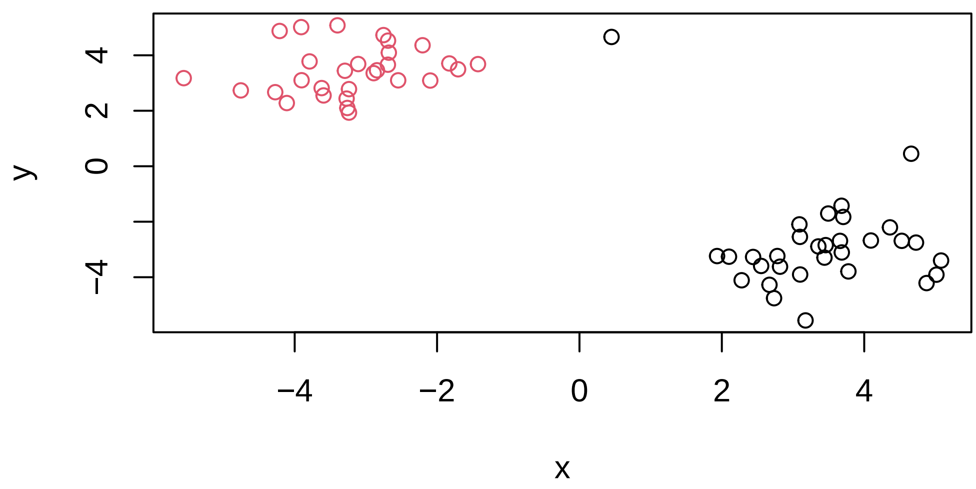
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3 3 3 3
[39] 3 3 3 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
```

```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 1 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.

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



#Principal Component Analysis (PCA)

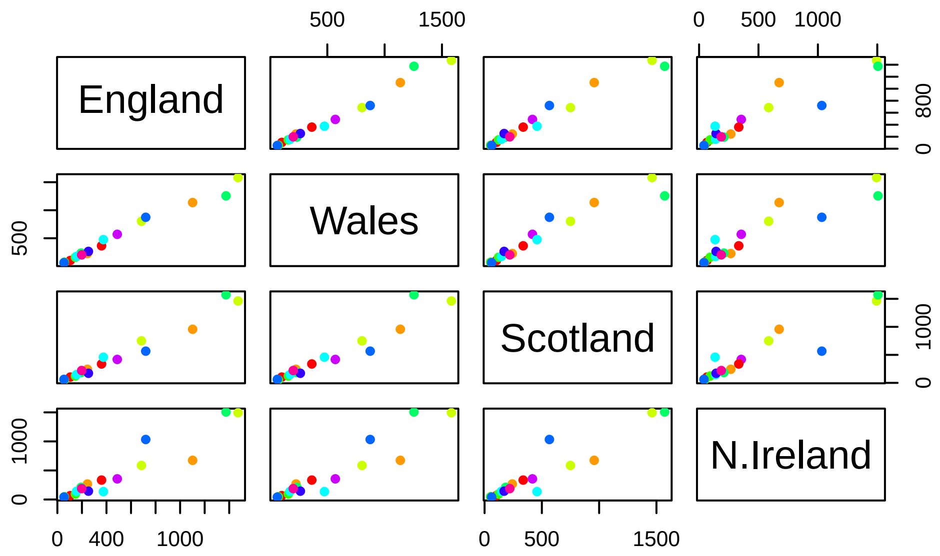
We will start PCA of tiny tiny dataset and make fun of stuff Barry eats.

```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
```

One useful plot in this case (because we only have 4 countries to look across) is so called pairs plot

```r
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 rows. It basecally wants the tranpoose pf the data we have

```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"
```

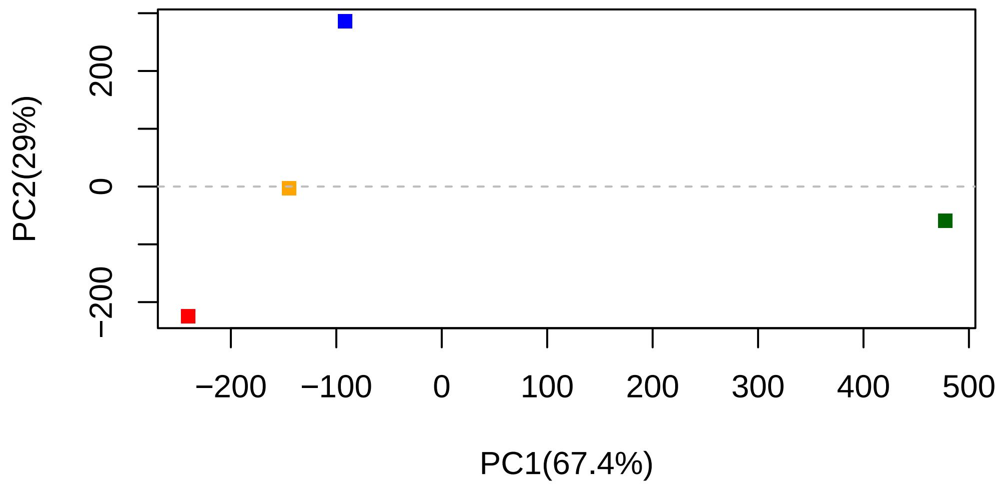
```r
pca$x
```

```
                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
```

```r
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
```

```r
plot(pca$x[,1], pca$x[,2], xlab = "PC1(67.4%)", ylab="PC2(29%)",
     col=c("orange", "red", "blue", "darkgreen"), pch=15)
abline(h=0, col="grey", lty=2)
abline(h=0, col="grey", lty=2)
```


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