Class 07: Machine Learning 1

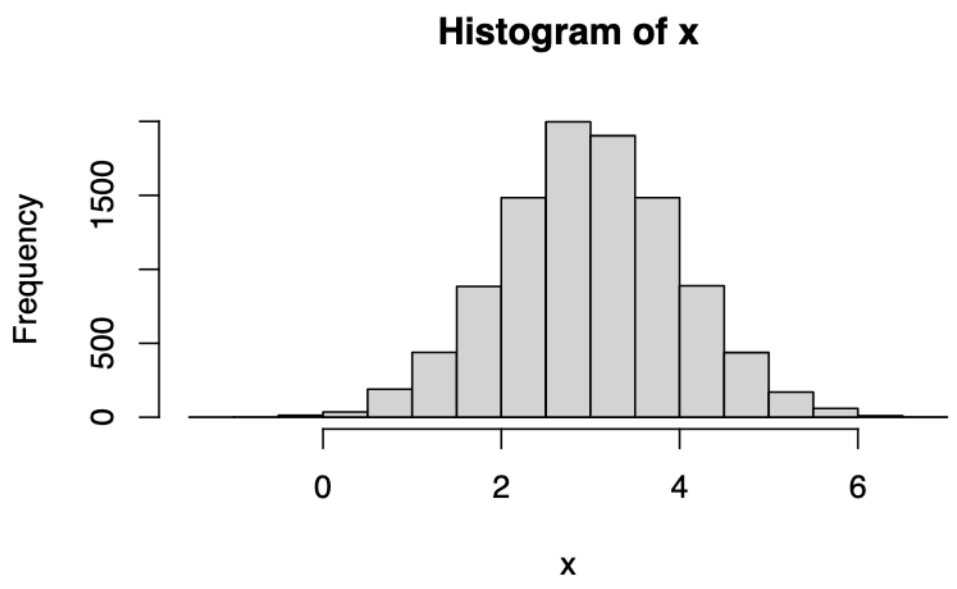
Kai Zhao(PID: A17599942)

Clustering

we will start today's lab with clustering methods, in particular so-called K-means. The main function for this in R is **`kmeans()`**

Let's try it on some made up data where we know that what the answer should be.

```r
x <- rnorm(10000, mean=3)
hist(x)
```



60 points

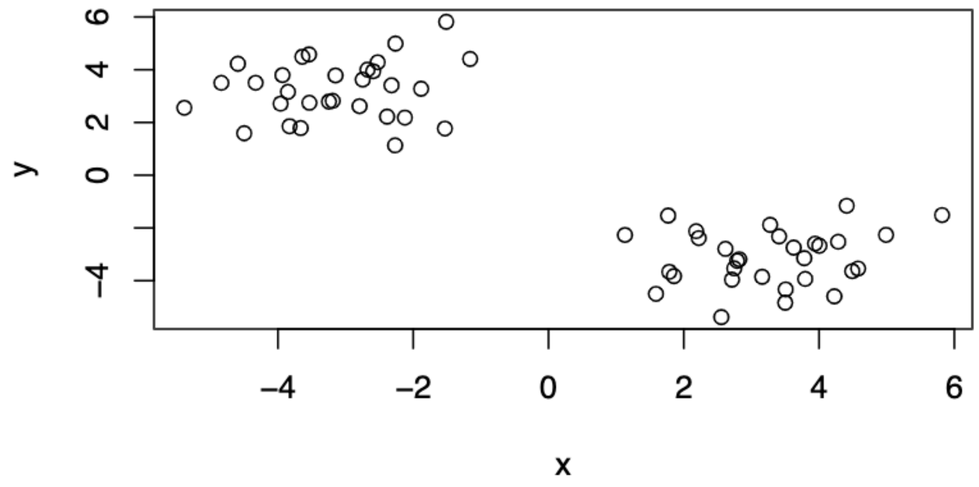
```r
tmp <- c(rnorm(30, mean=3), rnorm(30,mean=-3))
x <- cbind(x=tmp, y=rev(tmp))
x
```

```
              x         y
 [1,]  2.789181 -3.246384
 [2,]  1.785632 -3.664413
 [3,]  4.990367 -2.262505
 [4,]  4.004477 -2.675746
 [5,]  2.220653 -2.385723
 [6,]  2.820759 -3.189301
 [7,]  1.768562 -1.531468
 [8,]  3.499662 -4.836351
 [9,]  4.577516 -3.541133
[10,]  3.276770 -1.883271
[11,]  2.615736 -2.793107
[12,]  2.183321 -2.122602
[13,]  5.818104 -1.512527
[14,]  4.225951 -4.592156
[15,]  2.746062 -3.534720
[16,]  3.781665 -3.148096
[17,]  3.408777 -2.320842
[18,]  3.624595 -2.746581
[19,]  1.130785 -2.267443
[20,]  3.505646 -4.329642
[21,]  1.854807 -3.827730
[22,]  4.408753 -1.158637
[23,]  3.940465 -2.591902
[24,]  4.282357 -2.524787
[25,]  4.490019 -3.639081
[26,]  2.712705 -3.960268
[27,]  3.794480 -3.933003
[28,]  1.588639 -4.498693
[29,]  2.555489 -5.383949
[30,]  3.158643 -3.852315
[31,] -3.852315  3.158643
[32,] -5.383949  2.555489
[33,] -4.498693  1.588639
[34,] -3.933003  3.794480
[35,] -3.960268  2.712705
[36,] -3.639081  4.490019
[37,] -2.524787  4.282357
```

```
[38,] -2.591902  3.940465
[39,] -1.158637  4.408753
[40,] -3.827730  1.854807
[41,] -4.329642  3.505646
[42,] -2.267443  1.130785
[43,] -2.746581  3.624595
[44,] -2.320842  3.408777
[45,] -3.148096  3.781665
[46,] -3.534720  2.746062
[47,] -4.592156  4.225951
[48,] -1.512527  5.818104
[49,] -2.122602  2.183321
[50,] -2.793107  2.615736
[51,] -1.883271  3.276770
[52,] -3.541133  4.577516
[53,] -4.836351  3.499662
[54,] -1.531468  1.768562
[55,] -3.189301  2.820759
[56,] -2.385723  2.220653
[57,] -2.675746  4.004477
[58,] -2.262505  4.990367
[59,] -3.664413  1.785632
[60,] -3.246384  2.789181
```

We can pass this to the base R `plot()` function for a quick

```r
plot(x)
```



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

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

Cluster means:
          x         y
1  3.252019 -3.131813
2 -3.131813  3.252019

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] 67.90501 67.90501
 (between_SS / total_SS =  90.0 %)

Available components:

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

Q1. How many points are in each cluster

```r
k$size
```

```
[1] 30 30
```

Q2. Cluster membership?

```r
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
```

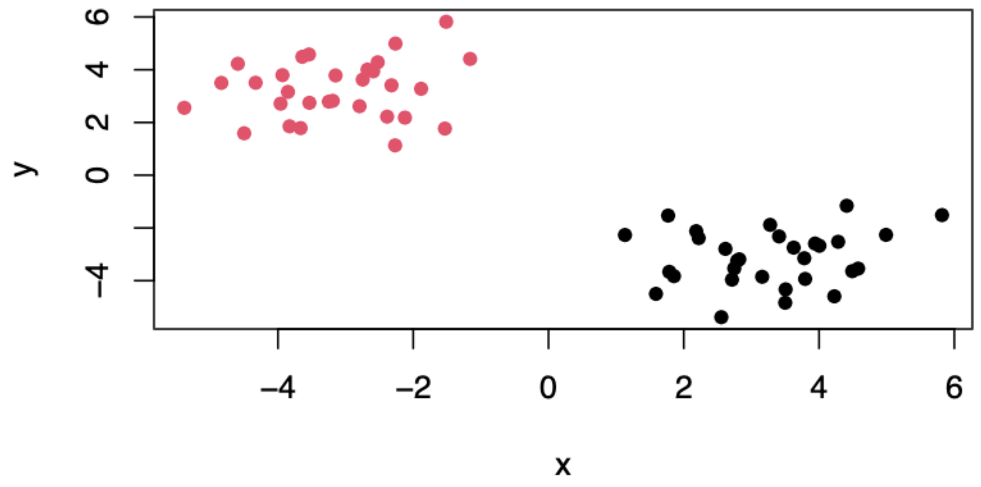
Q3. Cluster centers?

```r
k$centers
```

```
          x         y
1  3.252019 -3.131813
2 -3.131813  3.252019
```

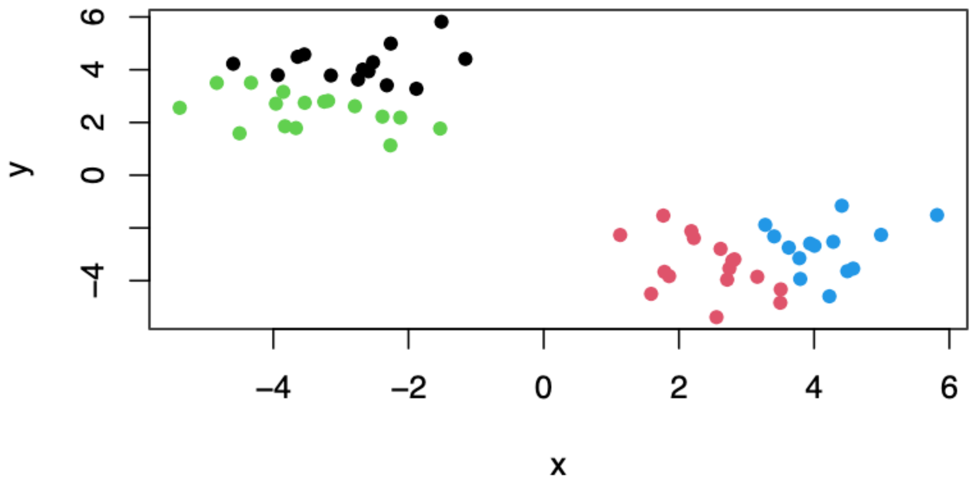
Q4 Plot my clustering results

```r
plot(x, col= k$cluster, pch=16)
```



Q5 Cluster the data again with kmean() into 4 groups and plot the results.

```r
k4 <- kmeans(x, center=4, nstart=20)
plot(x, col=k4$cluster, pch=16)
```



K-means is very popular mostly because it is fast and relatively straightforward to run and understand. It has a big limitation in that you need to tell it how many groups(k, or centers) you want.

hierarchical clustering

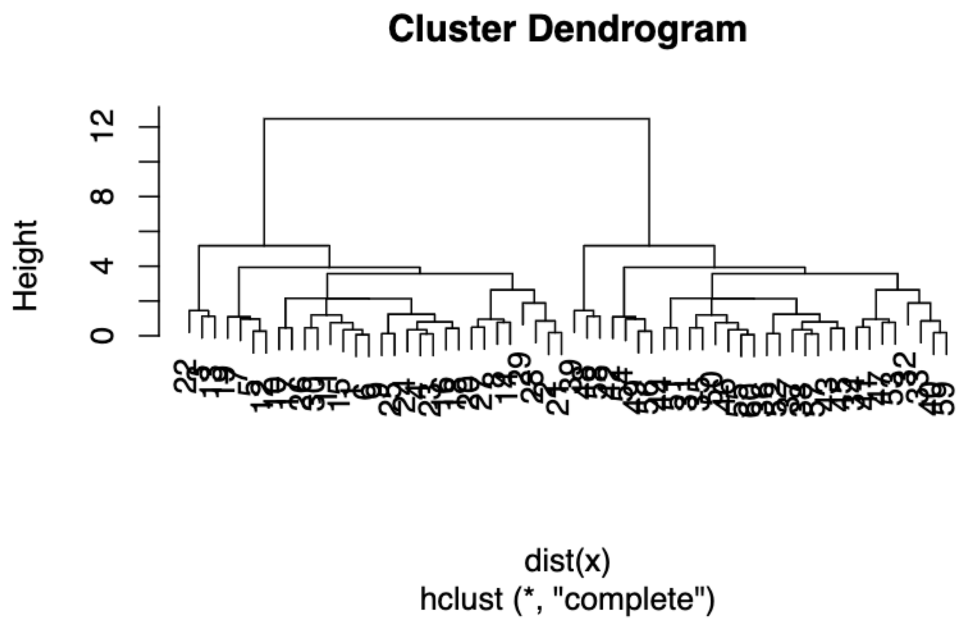
The main function in base R is called `hclust()`. You have to pass it in a "distance matrix" not just your input data.

You can generate a distance matrix with the `dist()` function.

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

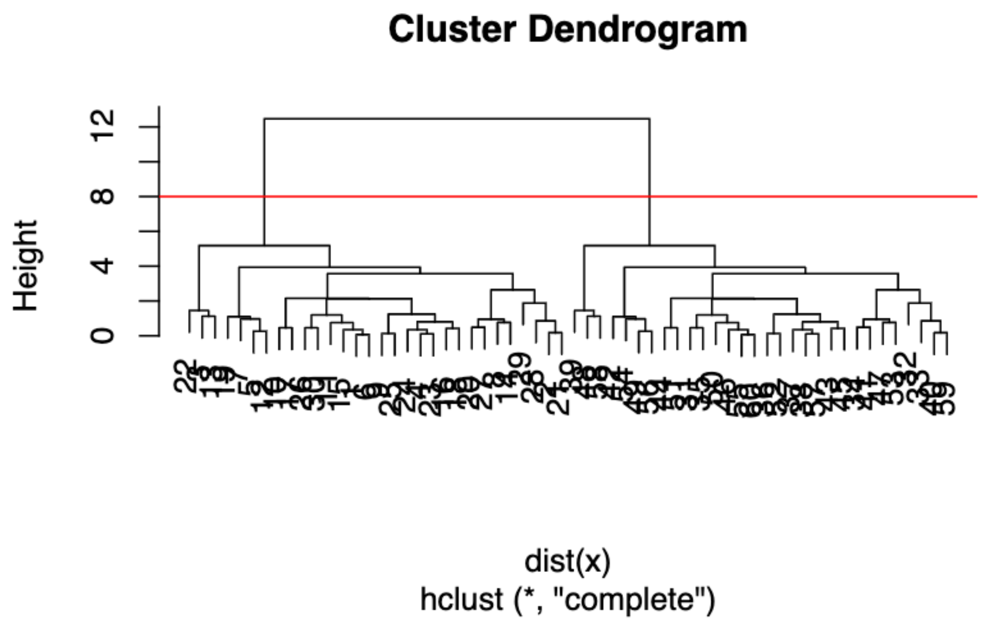
```

Call:
hclust(d = dist(x))

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

```
plot(hc)
```



To find the clusters (cluster membership vector) from a `hclust()` result we can "cut" the tree at a certain height taht we like. For this we use the `cutree()` function.

```
plot(hc)
abline(h=8,col="red")
```



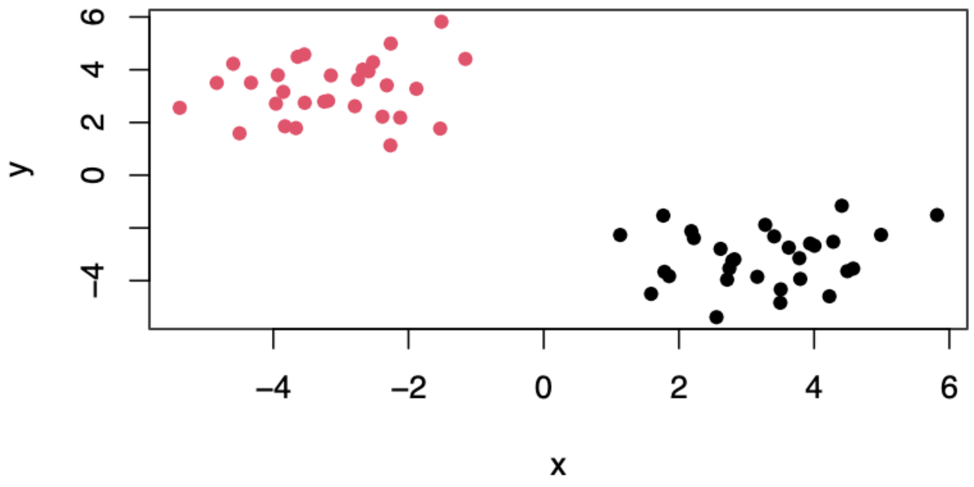
```r
grps <- cutree(hc, h=8)
```

```r
table(grps)
```

```
grps
 1  2 
30 30 
```

Q6 Plot our hclust results.

```r
plot(x, col=grps, pch=16)
```



Principal Component Analysis

PCA of UK food data

Read data showing the consumption in grams(per person, per week) of 17 different types of food-stuff measured and averaged in the four countries of the United Kingdom.

Let's see how PCA can help us but first we can try conventional analysis.

```
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
```

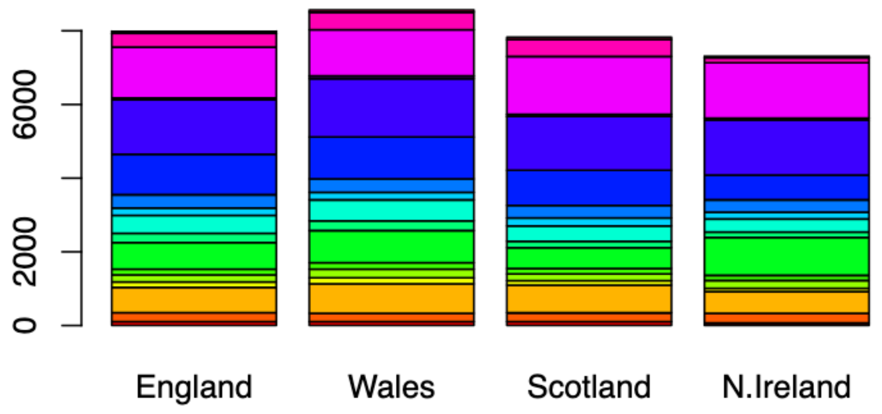
I need to fix that first column…

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

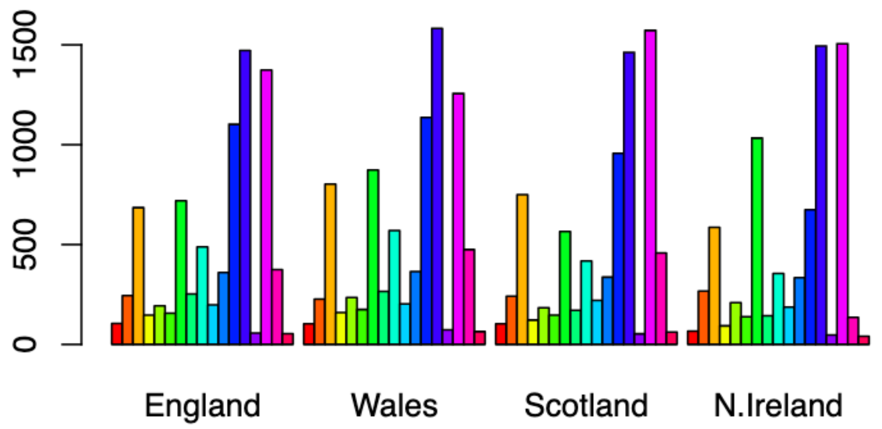
```
[1] "Cheese"          "Carcass_meat "   "Other_meat "     "Fish"
[5] "Fats_and_oils "  "Sugars"
```

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

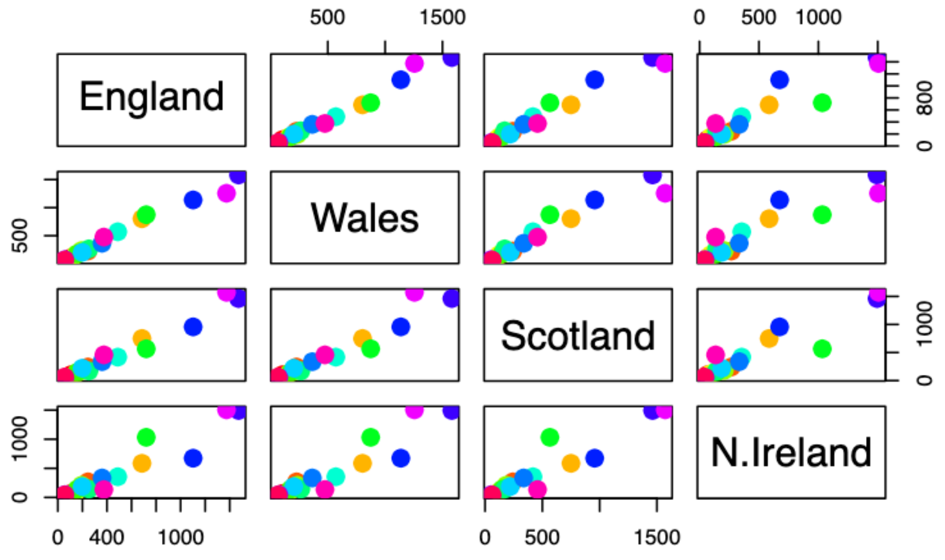
```r
cols <- rainbow(nrow(x))
barplot(as.matrix(x), col=cols)
```



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



```r
pairs(x, col=rainbow(17), pch=16, cex=2)
```



Principal Component Analysis (PCA)

PCA can help us make sense of these types of datasets. Ket's see how it works.

The main function in "base" R is called `prcomp()`. In this case we want to first take the transpose `t()` of our input `x` so the columns are the food types and the countries are the rows.

```r
head(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
```

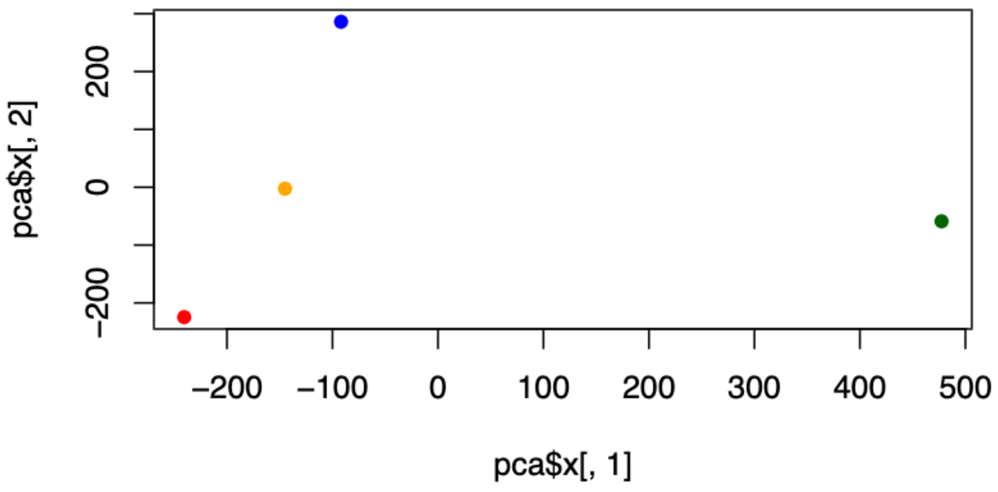
```
pca <- prcomp(t(x))
summary(pca)
```

```
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
```

```
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
```

```
plot( pca$x[,1], pca$x[,2], col=c("orange", "red", "blue", "darkgreen"), pch=16)
```

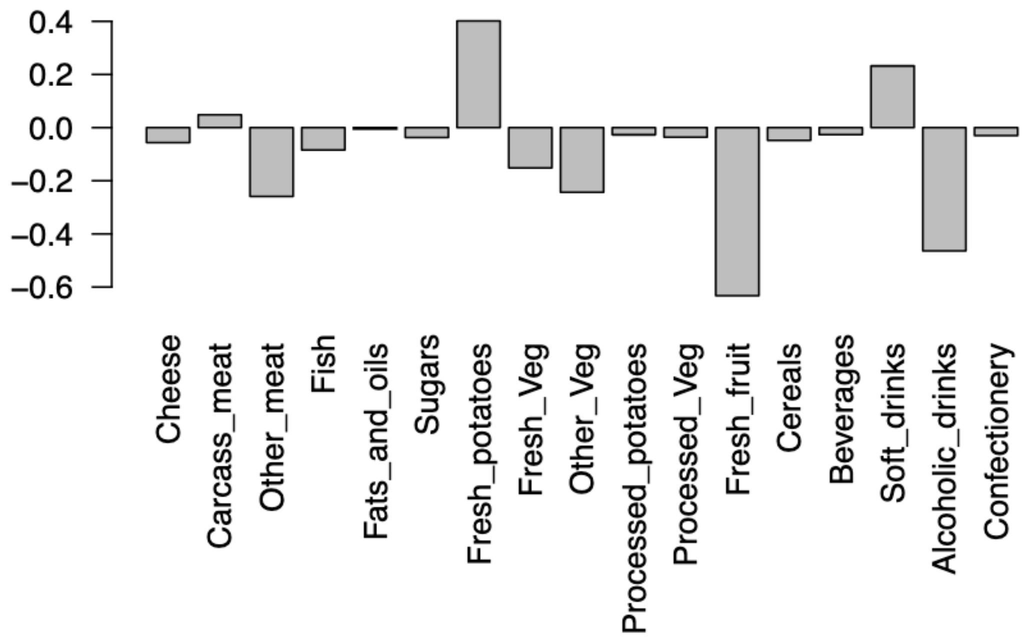


The "loadings' tell us how much the origional variables (in our case the food) contribute to the new variables i.e. the PCs.

```
head(pca$rotation)
```

```
                        PC1         PC2         PC3          PC4
Cheese         -0.056955380  0.01601285  0.02394295 -0.694538519
Carcass_meat    0.047927628  0.01391582  0.06367111  0.489884628
Other_meat     -0.258916658 -0.01533114 -0.55384854  0.279023718
Fish           -0.084414983 -0.05075495  0.03906481 -0.008483145
Fats_and_oils  -0.005193623 -0.09538866 -0.12522257  0.076097502
Sugars         -0.037620983 -0.04302170 -0.03605745  0.034101334
```

```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```


0.4
0.2
0.0
−0.2
−0.4
−0.6
Cheese
Carcass_meat
Other_meat
Fish
Fats_and_oils
Sugars
Fresh_potatoes
Fresh_Veg
Other_Veg
Processed_potatoes
Processed_Veg
Fresh_fruit
Cereals
Beverages
Soft_drinks
Alcoholic_drinks
Confectionery

Stop at Q9