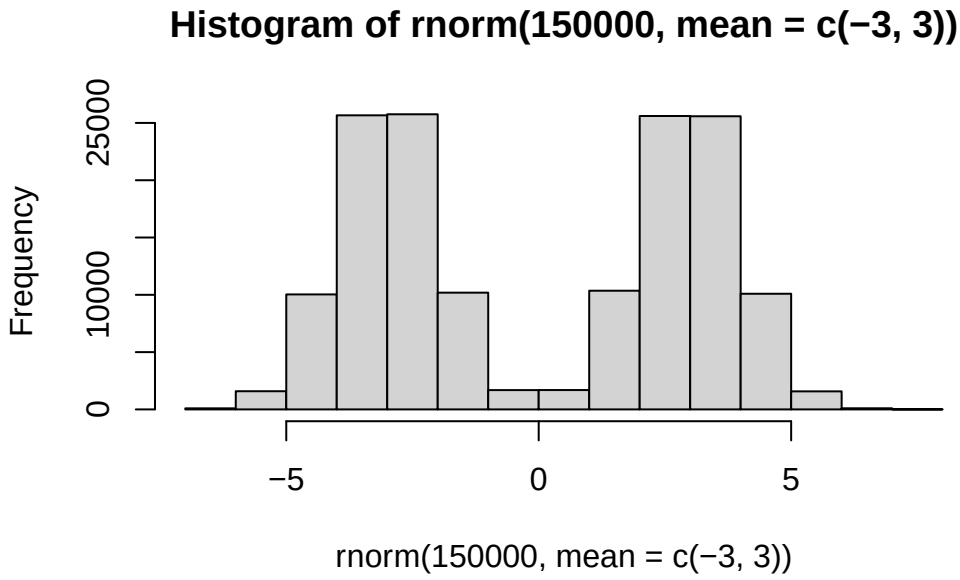
lab 7 Sydney Ackermann PID: A69036053

Machine learning k clustering

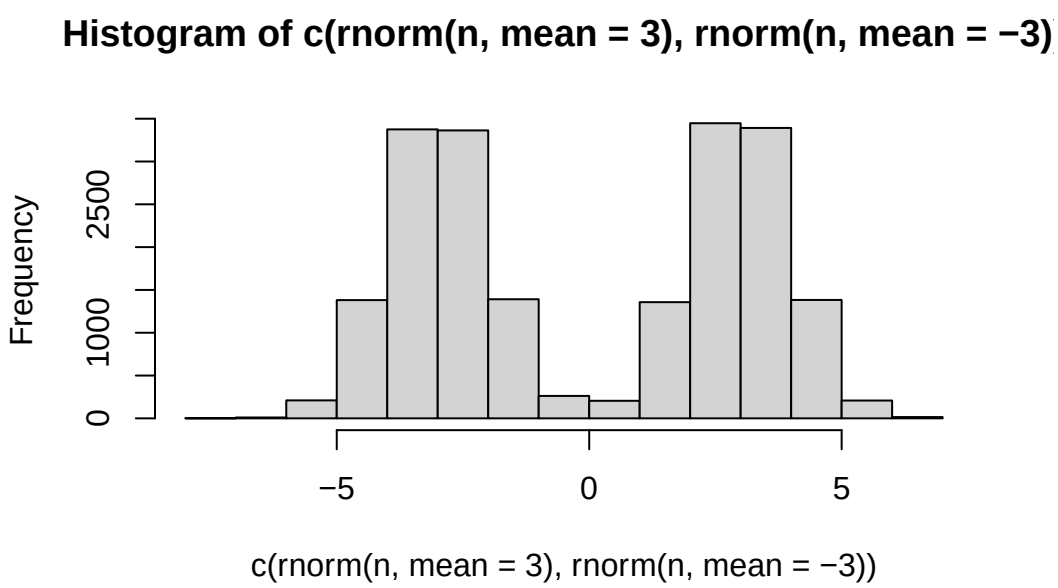
Before we get into clustering methods, let's make some sample data to cluster where we know what the answer should be.

To help with this I will use the `rnorm()` function

```
hist(rnorm(150000, mean=c(-3,3)))
```




```
n=10000
hist(c(rnorm(n,mean=3), rnorm(n, mean=-3)))
```



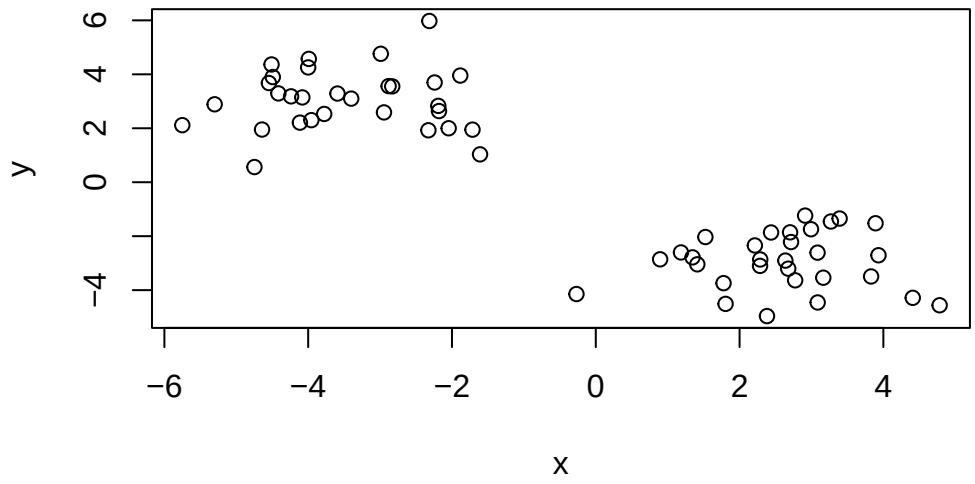

```
n=30
x<-(c(rnorm(n,mean=3), rnorm(n, mean=-3)))
#x
y<-(c(rnorm(n,mean=-3), rnorm(n, mean=3))) # rev(x)
#y
z<-cbind(x,y) # combine the vectors column bind (row bind would combine them row wise)
z
```

```
              x          y
 [1,]  4.4087499 -4.2856292
 [2,]  1.5246857 -2.0311461
 [3,]  1.1851252 -2.6030461
 [4,]  2.6367669 -2.9052083
 [5,]  2.9937040 -1.7426570
 [6,]  1.7773005 -3.7417790
 [7,]  2.6766973 -3.2048384
 [8,]  3.3916602 -1.3462065
 [9,]  1.3460344 -2.7853556
[10,]  2.9117636 -1.2376231
[11,] -0.2687086 -4.1452693
[12,]  2.2874230 -2.8640880
[13,]  3.1644225 -3.5385161
[14,]  2.7025201 -1.8576656
```

```
[15,]  3.8295178 -3.4937674
[16,]  3.2704286 -1.4559323
[17,]  2.2126681 -2.3438381
[18,]  4.7826002 -4.5610931
[19,]  3.9320692 -2.7078747
[20,]  0.8949868 -2.8573251
[21,]  2.4380749 -1.8635495
[22,]  2.2848051 -3.1013474
[23,]  2.7732775 -3.6365750
[24,]  1.4122218 -3.0435078
[25,]  1.8052072 -4.5090654
[26,]  2.3813252 -4.9626136
[27,]  3.8911910 -1.5213390
[28,]  3.0845696 -2.6105923
[29,]  3.0864345 -4.4570592
[30,]  2.7160519 -2.2167725
[31,] -1.6097295  1.0331562
[32,] -2.8809051  3.5629736
[33,] -2.1898539  2.8305954
[34,] -4.1151208  2.2089779
[35,] -2.3148087  5.9769022
[36,] -2.8309873  3.5533797
[37,] -1.7144122  1.9496342
[38,] -4.4142121  3.2931217
[39,] -4.5097750  4.3652389
[40,] -3.9554990  2.3012151
[41,] -3.7771510  2.5306567
[42,] -2.1806397  2.6380379
[43,] -2.0459893  2.0011866
[44,] -4.0814395  3.1437014
[45,] -4.4923117  3.8989030
[46,] -3.9998971  4.2559232
[47,] -4.7474307  0.5628564
[48,] -2.3278477  1.9258201
[49,] -5.3006988  2.8883488
[50,] -5.7510664  2.1155522
[51,] -2.2414756  3.6974435
[52,] -1.8845330  3.9539593
[53,] -3.5927613  3.2870878
[54,] -4.6415945  1.9530856
[55,] -4.5461229  3.6759716
[56,] -3.9914670  4.5712670
[57,] -3.4011066  3.0985748
```

```
[58,] -2.9456509  2.5881825
[59,] -4.2378817  3.1829415
[60,] -2.9904607  4.7603648
```

```
plot(z)
```



k-means clustering

The function in base R for k-clustering is called `kmeans()`.

```
km <- kmeans(z, centers=2)
#n=30 so its telling us there are 30 points in each group
# clustering vector is like a membership vector - tells us which cluster each point is closes
km
```

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

Cluster means:
          x         y
1  2.584452 -2.921043
2 -3.457094  3.060169

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 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

Within cluster sum of squares by cluster:
[1] 66.23869 75.70259
 (between_SS / total_SS =  88.4 %)

Available components:

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

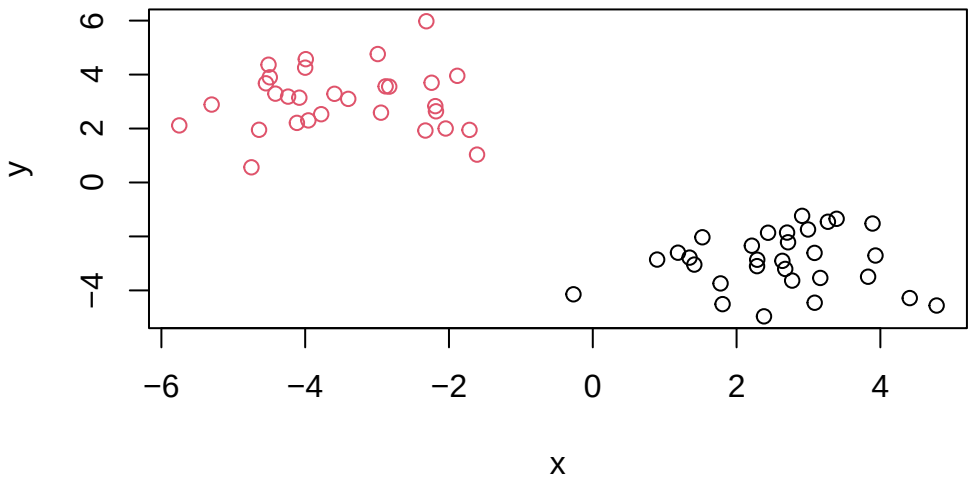
Q: print out the cluster membership vector

```
km$cluster
```

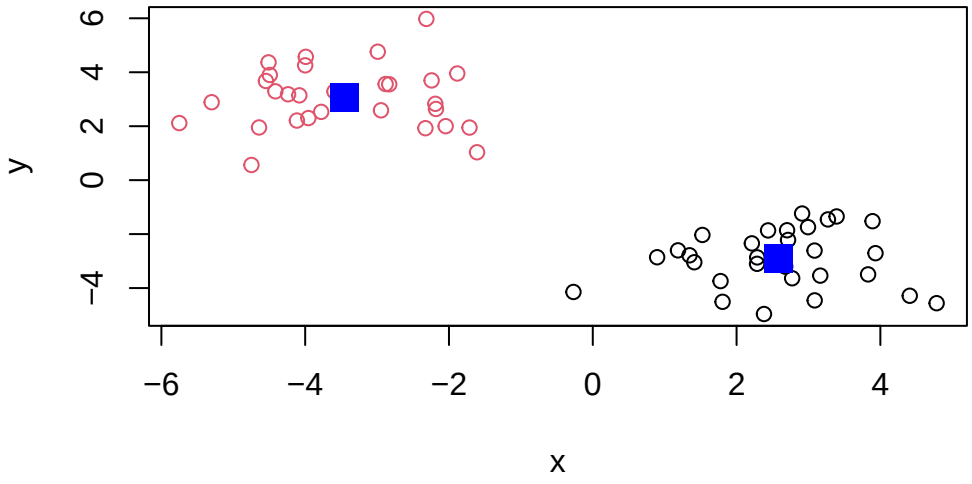
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Plot with clustering result

```
#plot(z, col= 1)
#plot(z, col= c("red", "blue"))
#plot(z, col= c(1,2))
# set color equal to the cluster membership vector:
plot(z, col= km$cluster)
```



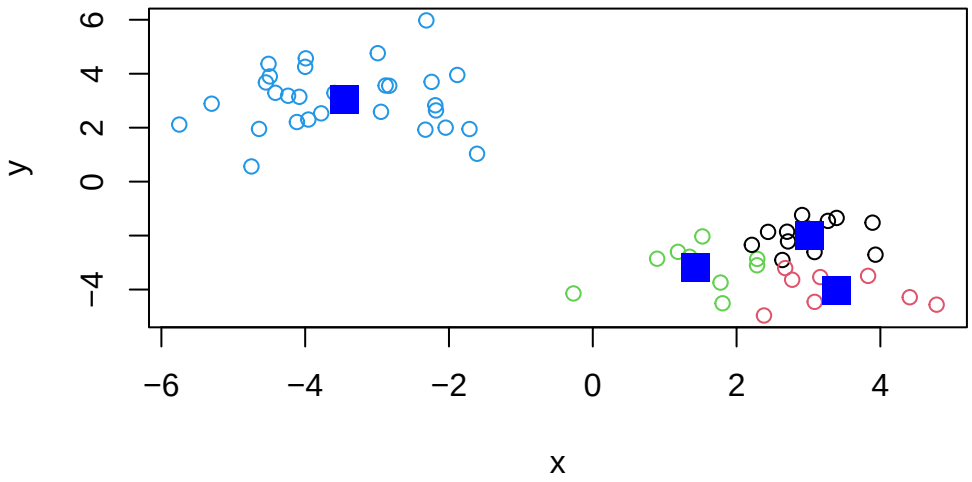

```
plot(z, col= km$cluster)
points(km$centers, col="blue", pch=15, cex=2) # cex makes point bigger if larger than 1, pch
```



```
# in ggplot you have to add layers. here it just does it automatically

# always have to specify number of clusters
```

Q: Can you cluster our data in `z` into four clusters please?

```
km4 <- kmeans(z, centers=4)
plot(z, col= km4$cluster)
points(km4$centers, col="blue", pch=15, cex=2)
```




```
# This is stochastic so it will be different every time
# kmeans will impose a structure (clustering) on your data even if its not there because you
```

Hierarchical clustering The main function to do hierarchical clustering in base R is called `hclust()`.

Unlike `kmeans()`. I can not just pass in my data as input, I first need a distance matrix (distance between each points (60x60 and zeros down the diagonal)).

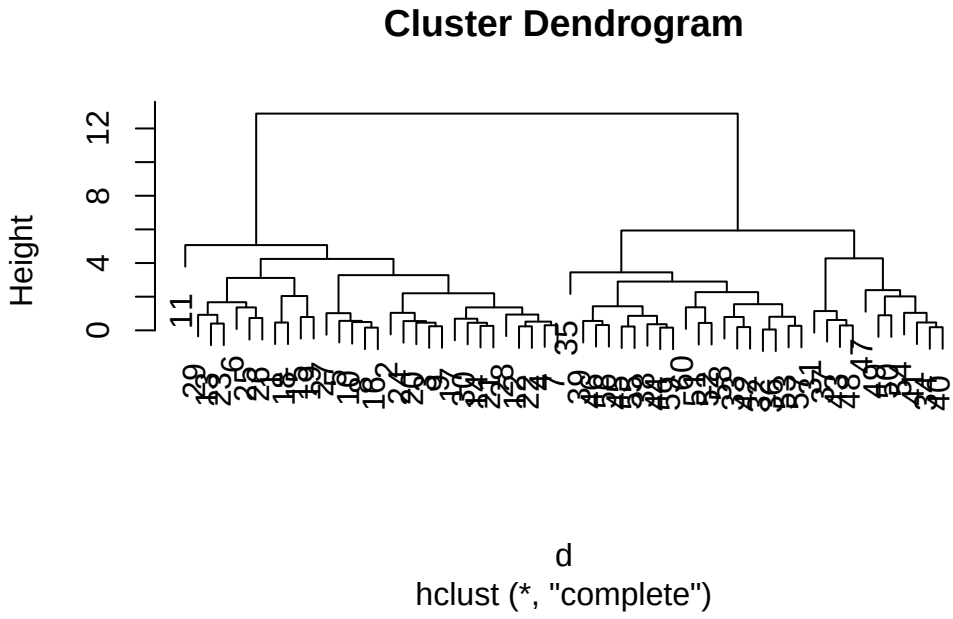
```
d<-dist(z) # make the distance matrix
hc<- hclust(d)
hc
```

```
Call:
hclust(d = d)

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

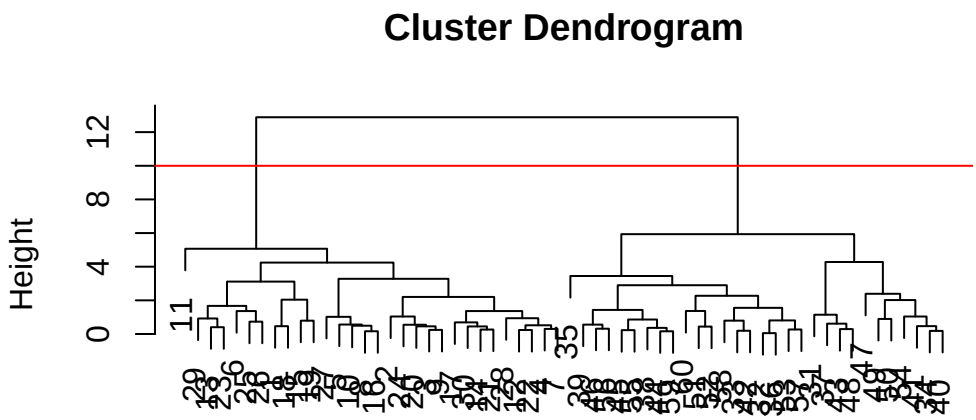
```
# bottom up - start with 60 clusters and then merge them untill we've stuck everything into c
# or top down clustering
```

There is a specific hclust plot() method...

```
plot(hc)
```




```
plot(hc)
abline(h=10, col="red") # adds line where the cluster number shows because the distance betwe
```



To get my clustering result (i.e., membership vector). I can "cut" my tree at a given height. To do this I will use the `cutree()`

```
grps<- cutree(hc, h=10) # this is our new vector of hierarchial clustering results
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 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
```

Principle component analysis (PCA)

Principal component analysis (PCA) is a well established "multivariate statistical technique" used to reduce the dimensionality of a complex data set to a more manageable number (typically 2D or 3D). This method is particularly useful for highlighting strong paterns and relationships in large datasets (i.e. revealing major similarities and diferences) that are otherwise hard to visualize. As we will see again and again in this course PCA is often used to make all sorts of bioinformatics data easy to explore and visualize.

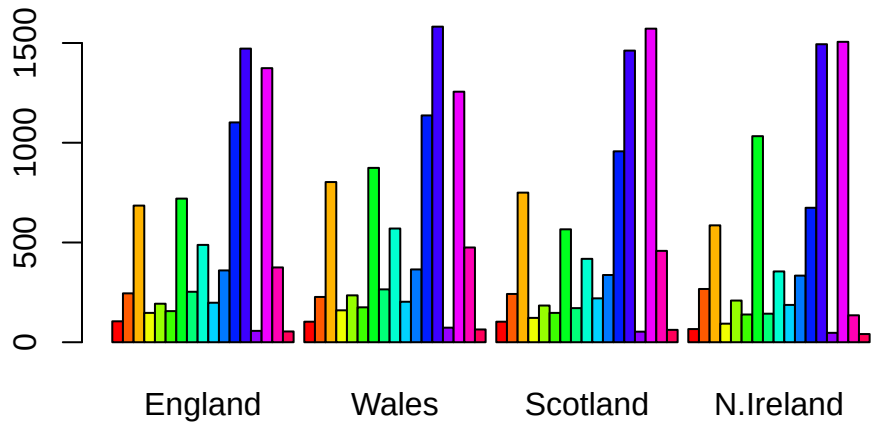
```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names=1) # make the columns the row names with =1
dim(x)
```

```
[1] 17  4
```

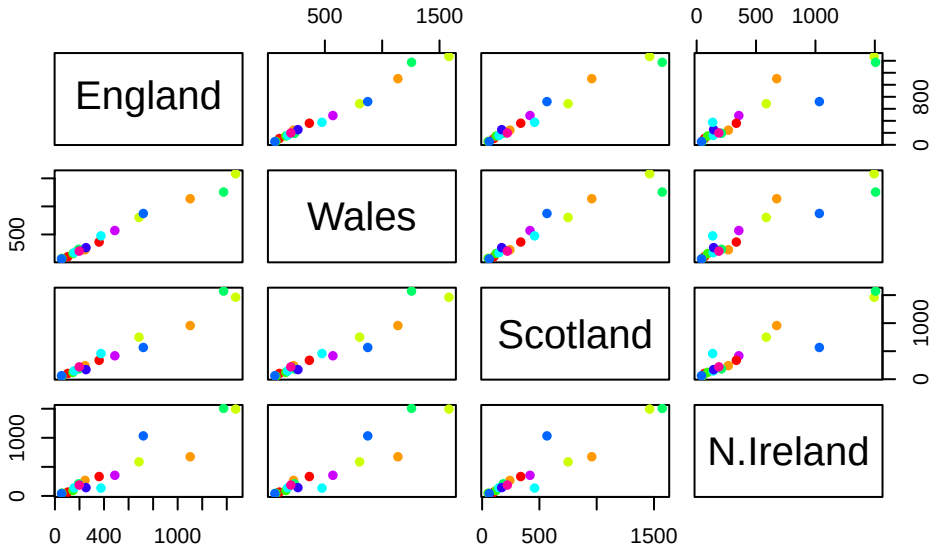
```
#View(x) always comment it unless you are looking at it

head(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
```

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




```
pairs(x, col=rainbow(10), pch=16)
```



PCA to the rescue

The main function to do PCA in base R is called `prcomp()`.

Note thatt I need to take the transpose of this particular data as that is what the `precomp()` help page was asking for.

```
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
```

```
# want to switch columnns and rowes so its accepted into PCA

t(x) # transpose
```

```
          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) # pc1 captures 67% of the variation in the data. pc2 captures 29% of the data
```

```
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
```

```
# (67+29=96, therefore, )96% of the varience was captured in 2 dimensions
```

Lets see what is inside our result object `pca` that we just calculated:

```
attributes(pca)
```

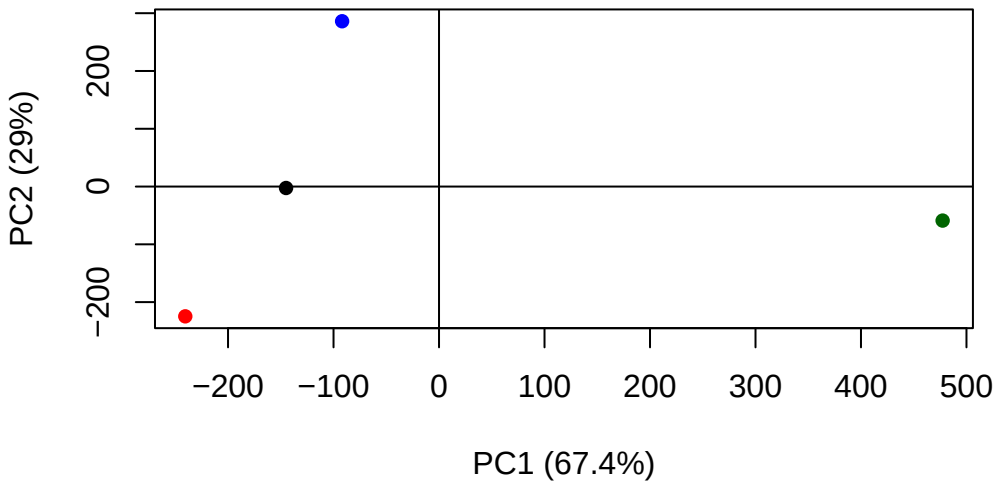
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```
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
```

To make our main result figure, called a "PC plot" (or "score plot" or "coordination plot" or "PC1 vs PC2 plot").

```
plot(pca$x[,1], pca$x[,2], col=c("black", "red", "blue", "darkgreen"), pch=16, xlab="PC1 (67
# pch fills in the dots

abline(h=0) # average
abline(v=0)
```

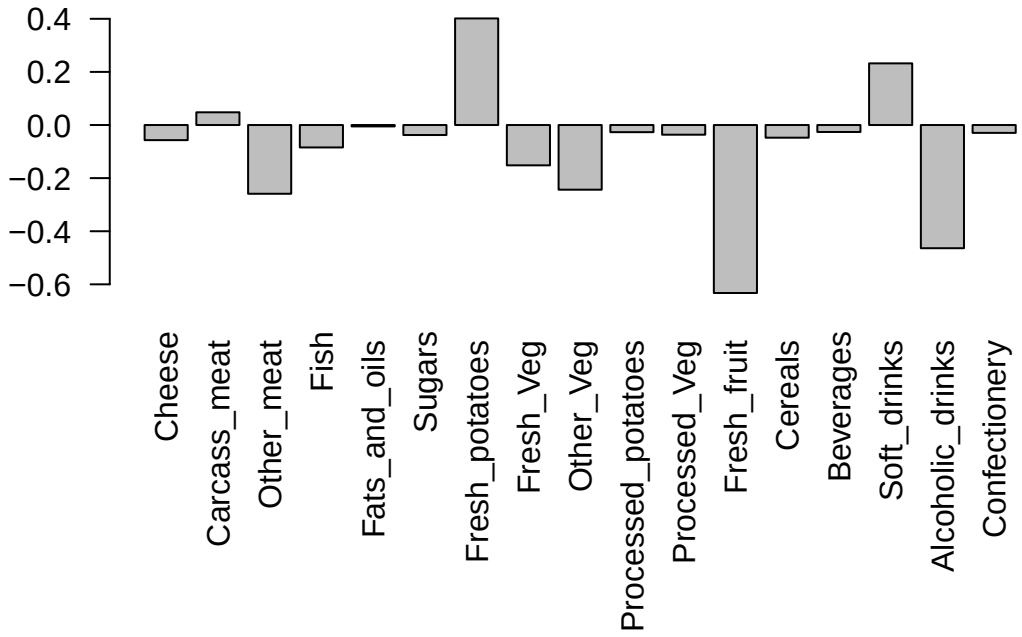


From here we can see that Ireland is off on it's own. The major axis of variance. Give more weight to PC1 because it captures more of the variance. If two points are far apart on PC1 then thats saying there is a major feature in this data. ireland is the main difference.

Variable loadings plot

Can give us insight on how the original variables (the foods) contribute to our new PC axis

```
## 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 )
```




```
# PC1 -> COMPARING TWO COUNTRIES
```

```
pca$rotations
```

```
NULL
```