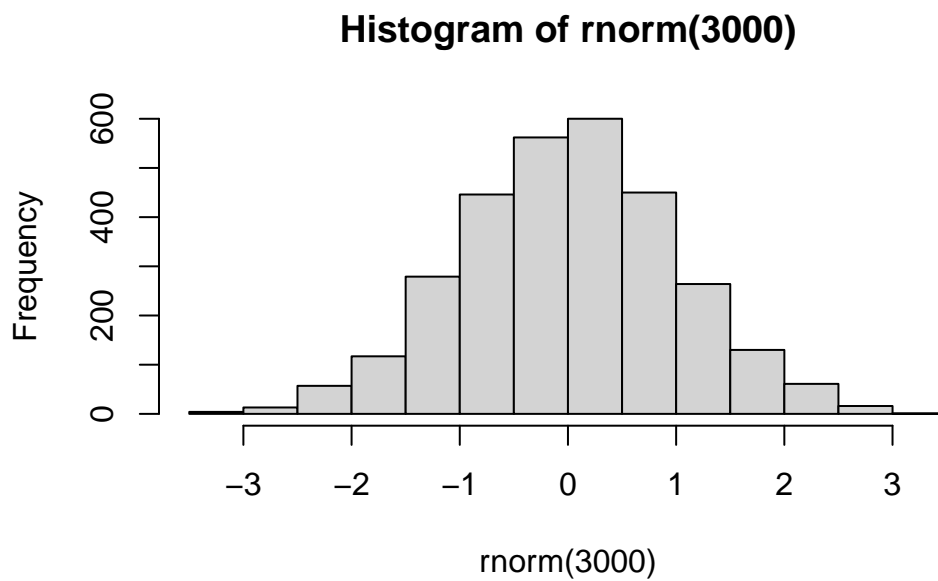# Class07

Katie Mostoller A17259578

Today we will delve into unsupervised machine learning with an initial focus on clustering and dimensionality reduction.

Let's starts by making up some data to cluster. The `rnorm()` function can help us here.

```
hist(rnorm(3000))
```
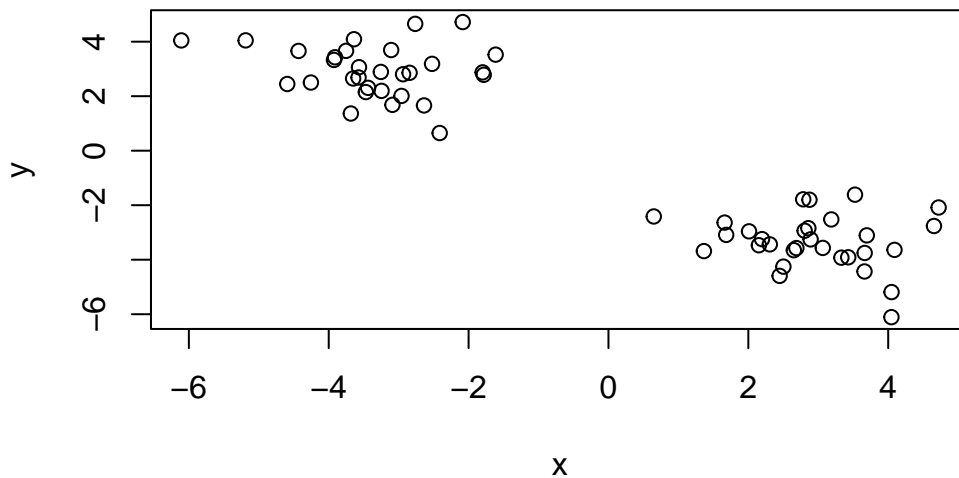
Histogram of rnorm(3000)

Lets get some data centeres at 3,-3

```
#Combine 30 +3 values with 30 -3 values

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

```
# Bind these values together
z <- cbind(x = x, y = rev(x))
head(z)
```

```
            x          y
[1,] 2.873860 -1.799017
[2,] 2.009860 -2.959311
[3,] 3.693799 -3.107143
[4,] 2.858311 -2.843940
[5,] 1.661003 -2.637604
[6,] 2.447735 -4.592990
```

```
plot(z)
```


K-means

Now we can see how K-means clusters this data. The main function for K-means clustering in "base R" is called `kmeans()`

2

```r
km <- kmeans(z, center = 2)
km
```

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

Cluster means:
          x         y
1 -3.341254  2.921022
2  2.921022 -3.341254

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] 54.67491 54.67491
 (between_SS / total_SS =  91.5 %)

Available components:

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

```r
attributes(km)
```

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

$class
[1] "kmeans"
```

Q. What size is each cluster?

```r
km$size
```

```
[1] 30 30
```

Q. The cluster membership vector (i.e. the answer: cluster to which each point is
allocated)

```
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 2 2 2 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 1
```
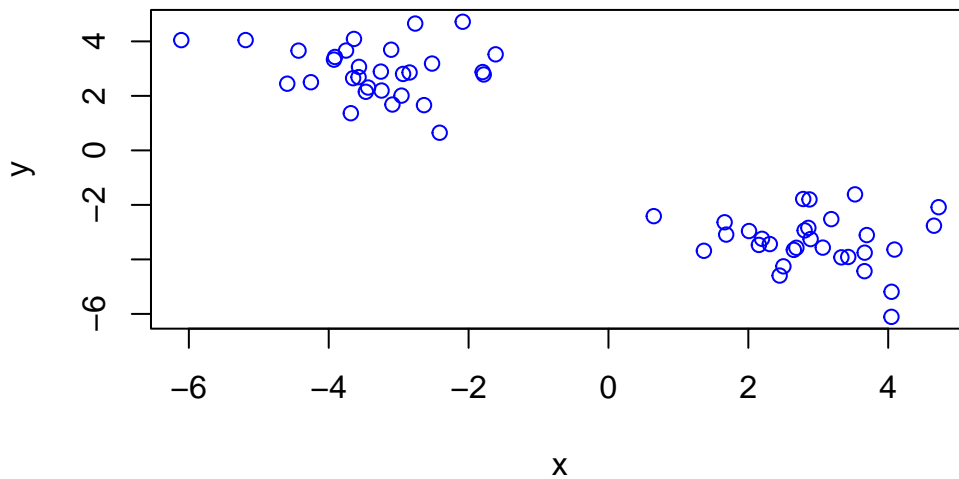
Cluster centers

```
km$centers
```

```
          x          y
1 -3.341254  2.921022
2  2.921022 -3.341254
```
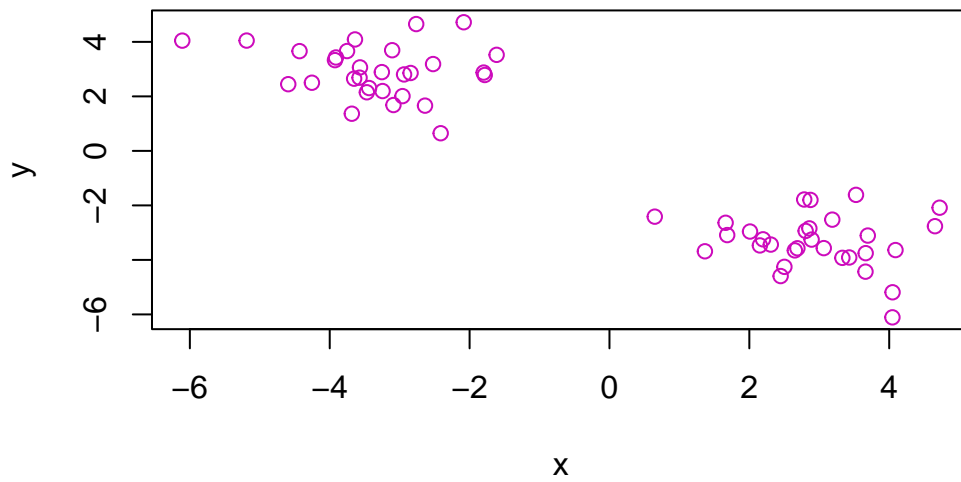
Q. Make a results figure, i.e. plot the data z colored by cluster membership and show the cluster centers.

```
plot(z, col = "blue")
```



```
plot(z, col = 6)
```
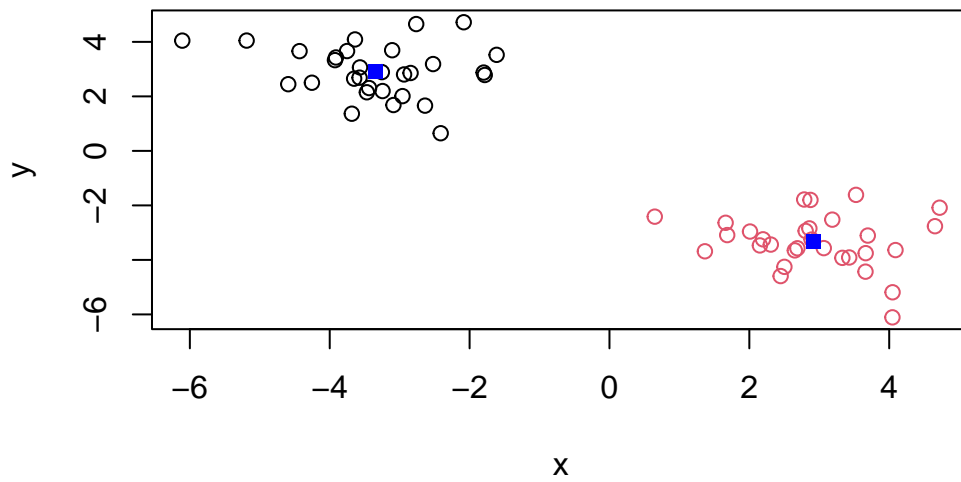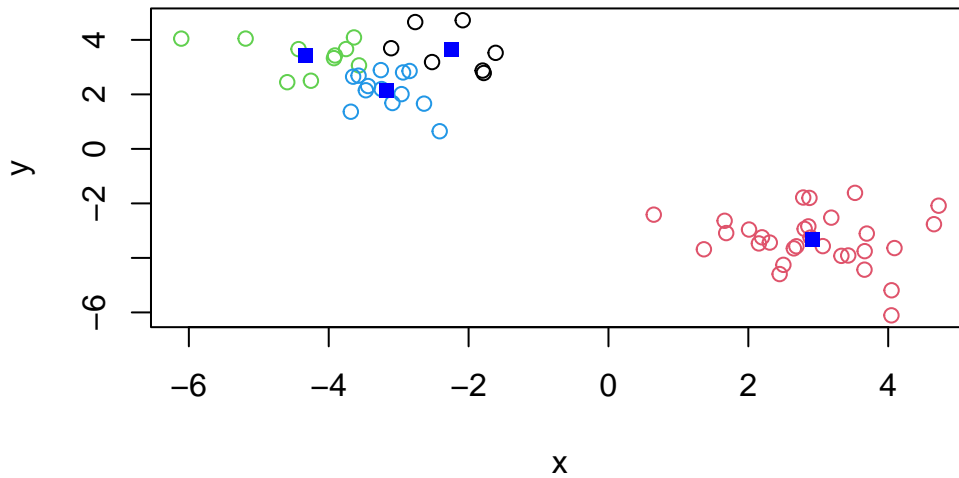
```
# Make each cluster a different color
plot(z, col = km$cluster)

# Add a point that represents the center of each cluster to the plot
points(km$centers, col = "blue", pch = 15)
```

Q. Re-run your k-means clsutering and ask for 4 cluster and plot the results as above

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

Hierarchical Clustering

The main "base R" function for this is `hclust()`. Unlike `kmeans()` you can't just give your dataset as input, you need to provide a distance matrix.

We can us the `dist()` function for this

```
d <- dist(z)
```

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

```
Call:
hclust(d = d)

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

There is a custom `plot()` for hclust objects. Let's see it.

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

Cluster Dendrogram

d
hclust (*, "complete")
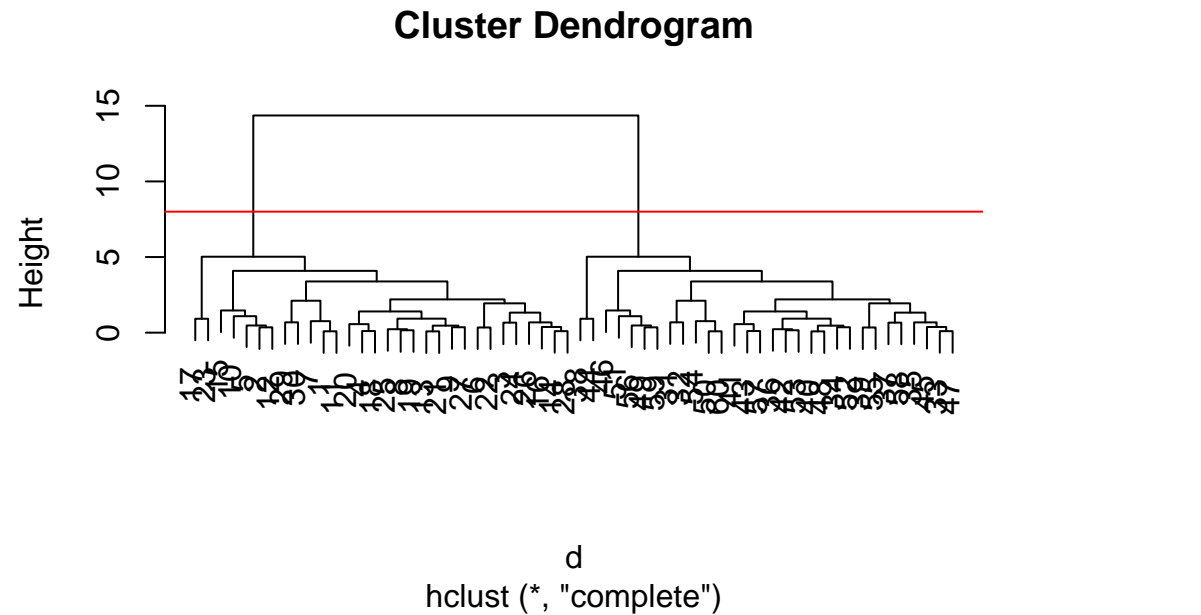
The function to extract clusters/groups from an hclust object/tree is called `cutree()`.

```
grps <- cutree(hc, h = 8)
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 data with hlcust clusters

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

```r
cutree(hc, k = 2)
```

```
 [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
```
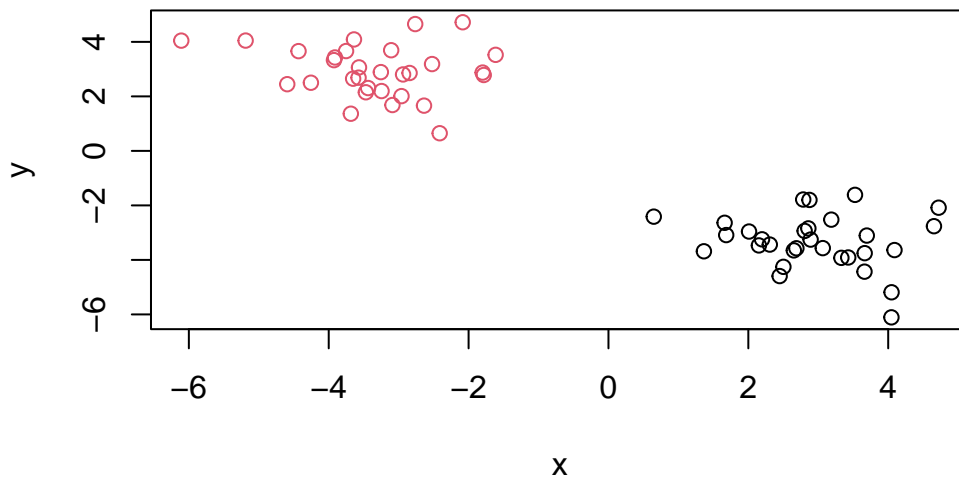
Principal Component Analysis (PCA)

the main function for PCA in base R for PCA is called `prcopm()`. There are many, many add on packages with PCA functions tailored to particular data types (RNSASeq, protein, structures, metagenomica, etc...)

PCA of UK food data

Read the data into R, it is a CSV file and we can use `read.csv()` to read it:

```r
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 would like the food names as row names not as their own column of data

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

A better way to do this is to do it at the same time of data import with `read.csv()`

```
food <-  read.csv(url, row.names =1)
food
```

	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

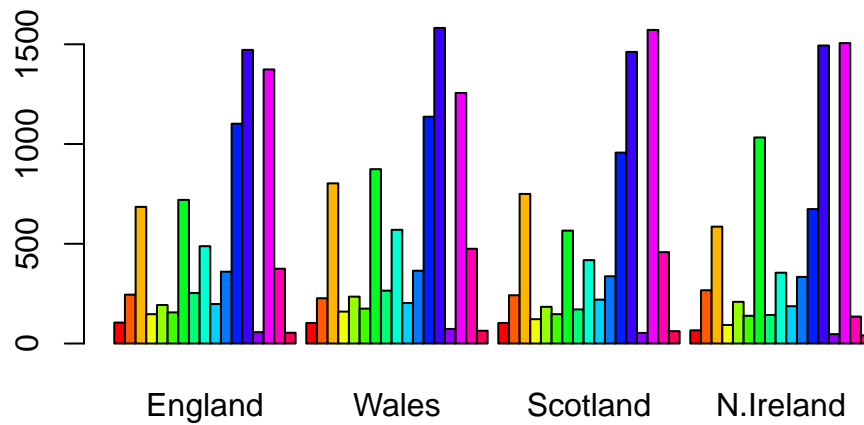Let's make some plots and dig into the data a little.

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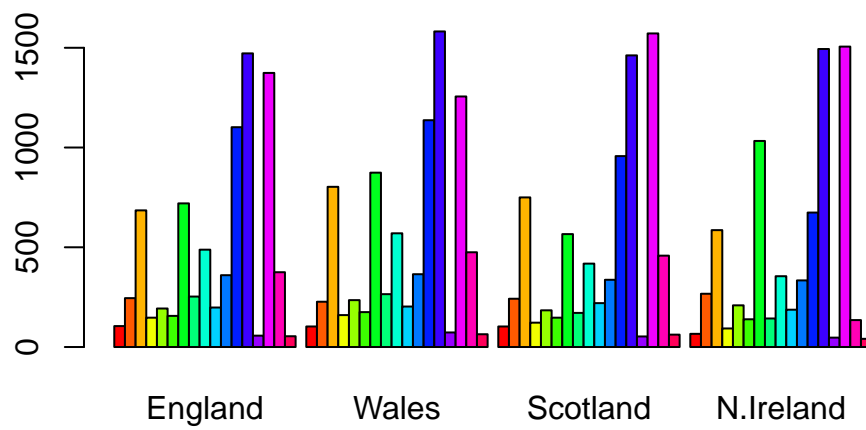
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(food)))
```
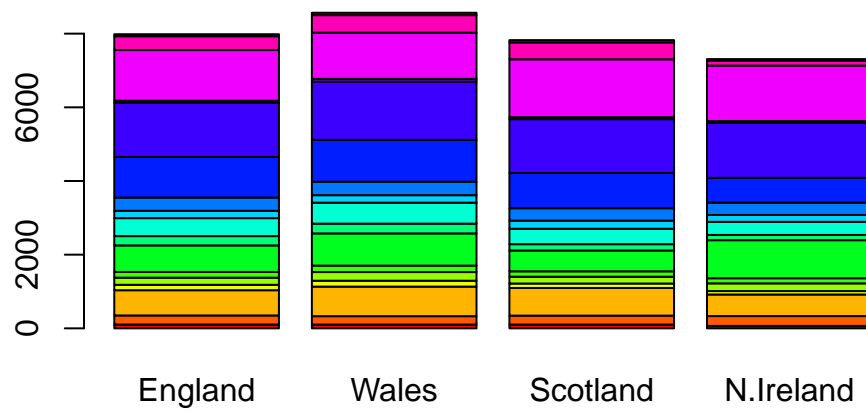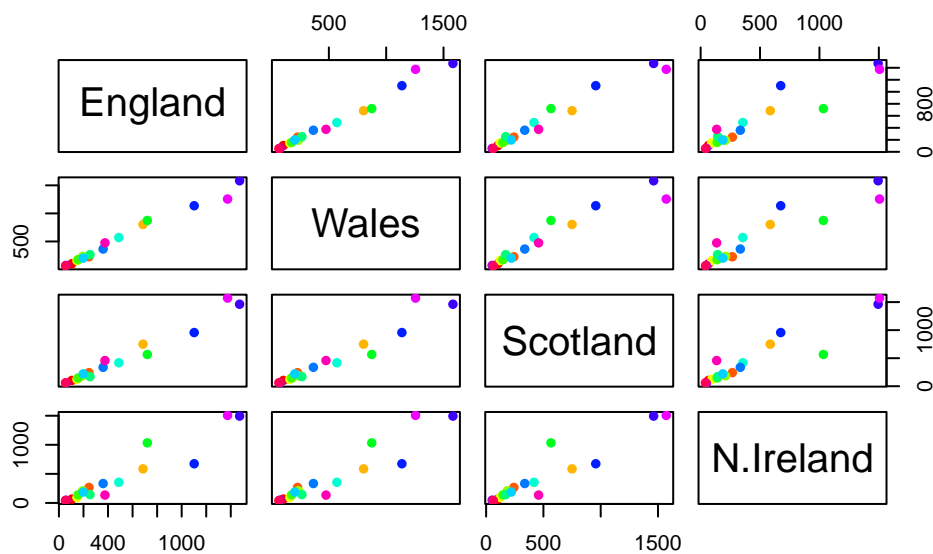
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(food)))
```


How about a "pairs" plot where we plot each country against all other countries.

```
pairs(food, col=rainbow(nrow(food)), pch=16)
```

Really there has to be a better way……

PCA to the rescue!

We can run a Principal Component Analysis (PCA). We ned to take the transpose to get the foods in the columns and the countries in the rows

```
pca <- prcomp(t(food))
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
```

What is in my `pca` results object?

```
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 my main result figure, often called a PC plot (or score olot, oridenation plot, or PC1 vs PC2 plot)

```
plot(pca$x[,1], pca$x[,2], xlab = "PC1", ylab = "PC2", col = c("orange", "red", "blue", "darl
```
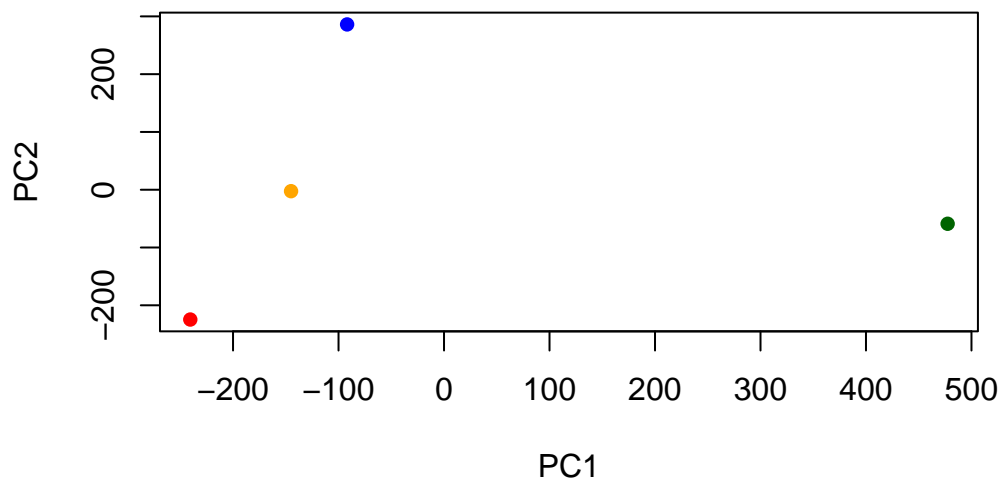

To see the contributions of the origional variables (foods) to these new PCs we can look at the `pca$rotation` component of our results

```
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
library(ggplot2)
loadings <-  as.data.frame(pca$rotation)

loadings$name <- rownames(loadings)

ggplot(loadings) + aes(PC1, name) + geom_col()
```
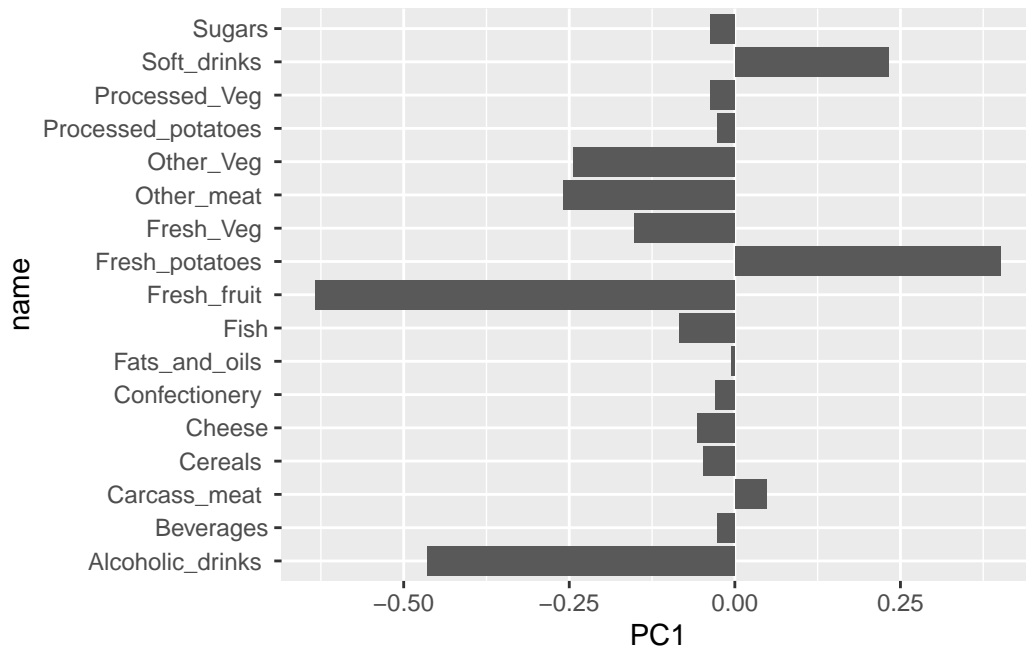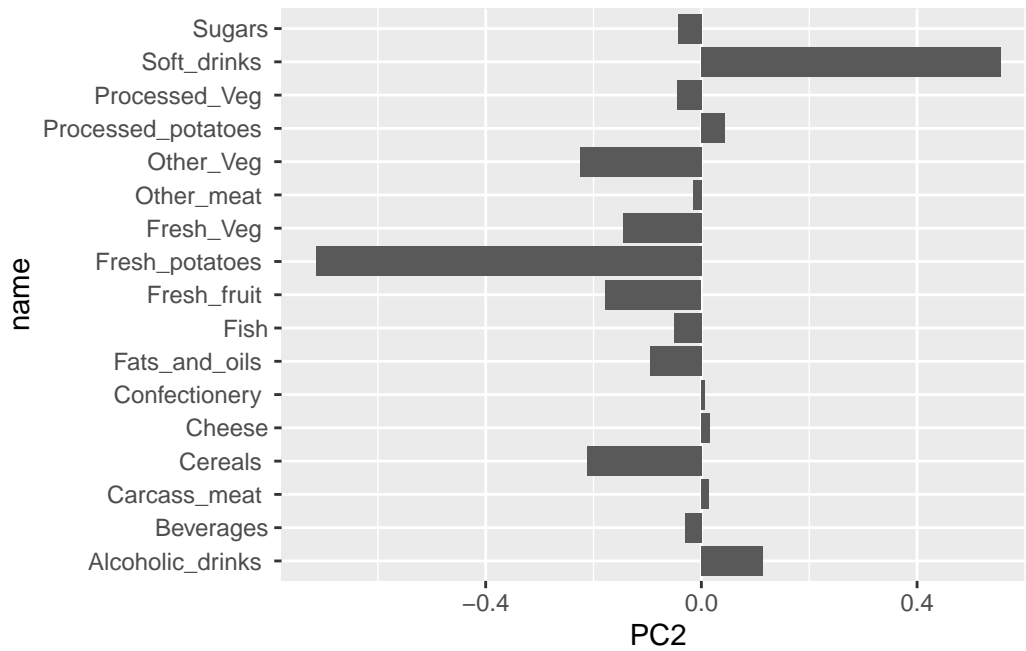
```
loadings <-  as.data.frame(pca$rotation)

loadings$name <- rownames(loadings)

ggplot(loadings) + aes(PC2, name) + geom_col()
```

```
loadings <- as.data.frame(pca$x)
loadings_lab <- tibble::rownames_to_column(loadings, "Country")

ggplot(loadings_lab) +
  aes(PC1, PC2, col=Country) +
  geom_point()
```