Lab 7

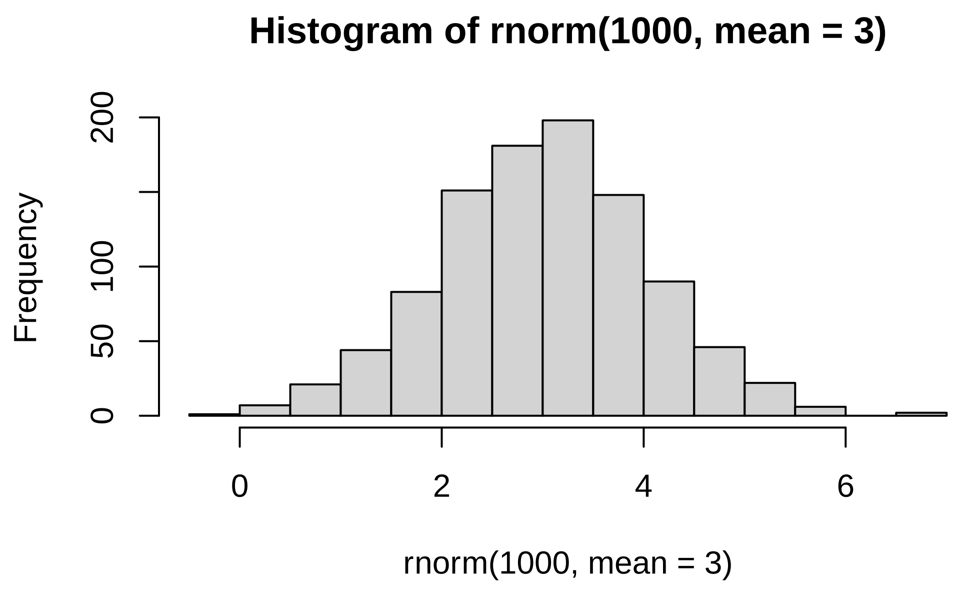
Gretel Warmuth (PID: A17595945)

Today we are going to learn how to apply different machne learning methods, beginning with clustering:

The goal is to find groups/clusters in the input data.

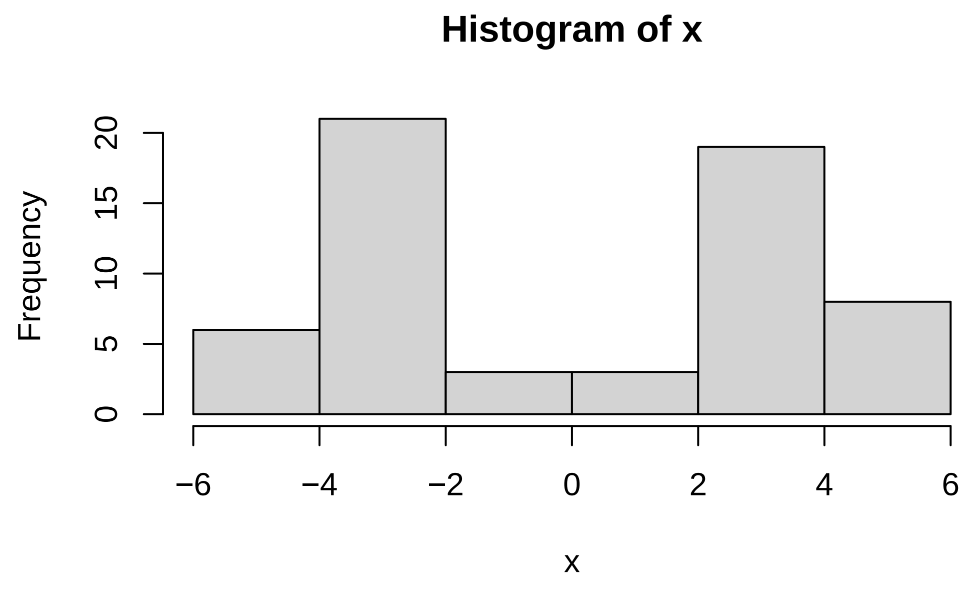
First I will make some random data with clear groups with the `rnorm()` function:

```
hist(rnorm(1000, mean=3))
```



Making a histogram with two peaks:

```
n <- 30
x <- (c(rnorm(n, -3), rnorm(n, +3)))
hist(x)
```

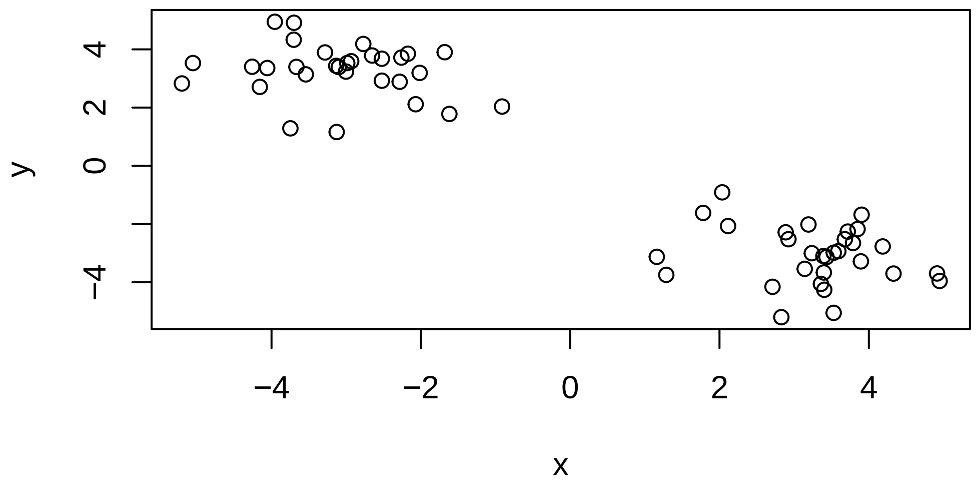


Making a cluster plot:

```r
n <- 30
x <- (c(rnorm(n, -3), rnorm(n, +3)))
y <- rev(x)

z<- cbind(x,y)
head(z)
```

```
             x        y
[1,] -1.617962 1.780855
[2,] -5.052637 3.528754
[3,] -2.522329 2.923445
[4,] -3.002405 3.236163
[5,] -3.699434 4.913742
[6,] -2.770901 4.185858
```

```r
plot(z)
```



Use the `kmeans()` function setting k to 2 and nstart=20

Inspect/print the results

> Q. How many points are in each cluster?
>
> Q. What 'component' of your result object details - cluster size? - cluster assignment/membership? - cluster center?
>
> Q. Plot x colored by the kmeans cluster assignment and add cluster centers as blue points

```
km <- kmeans(z, centers = 2)
km
```

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

Cluster means:
          x         y
1 -3.069540  3.271348
2  3.271348 -3.069540

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] 52.31094 52.31094
 (between_SS / total_SS =  92.0 %)

Available components:

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

Results of the object `km`

```
attributes(km)
```

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

$class
[1] "kmeans"
```

Cluster size:

```
km$size
```

```
[1] 30 30
```

Cluster assignment/membership:

```
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 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
```

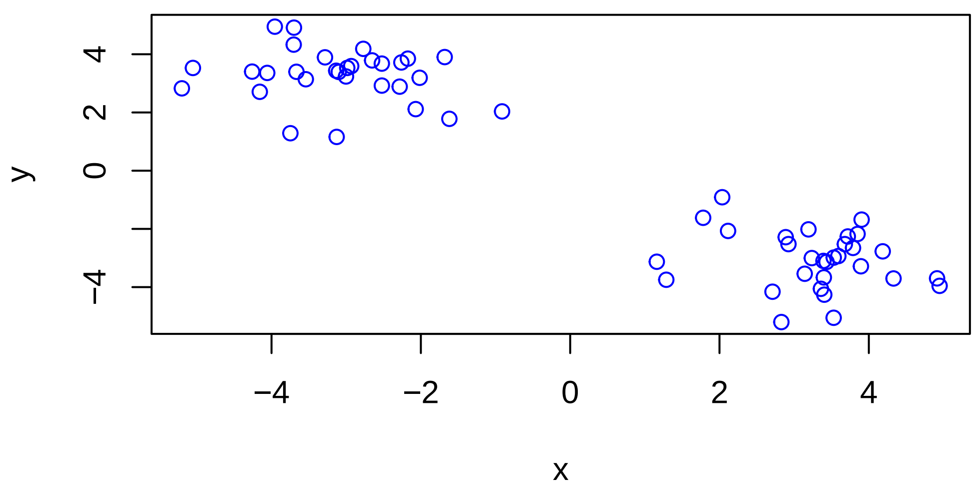
Cluster center:

```
km$centers
```

```
          x         y
1 -3.069540  3.271348
2  3.271348 -3.069540
```

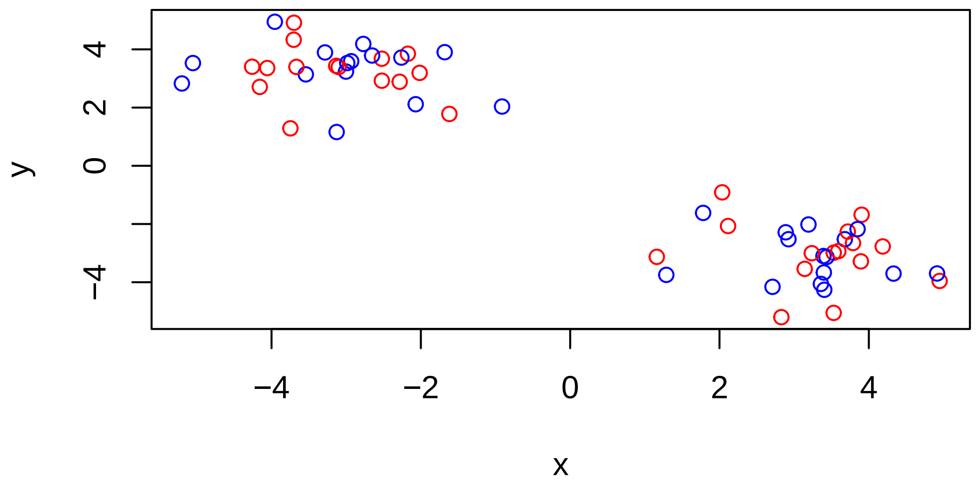
Plot:

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



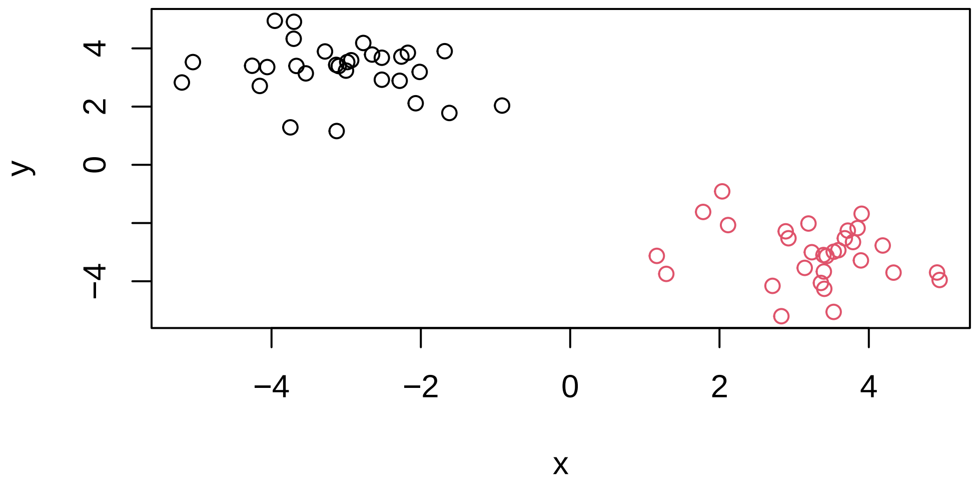
R will recycle the shorter color vector to be the same length as the longer (number of data points) in z

```r
plot(z, col=c("red", "blue"))
```



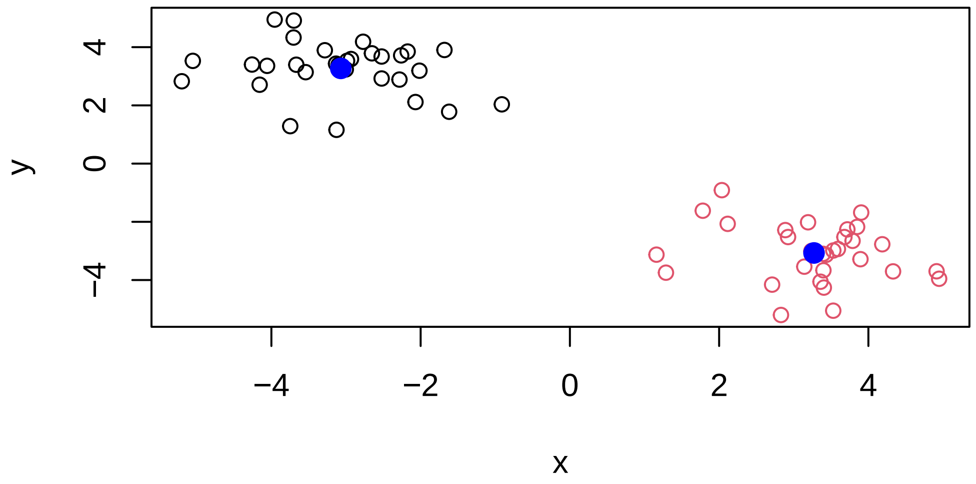
Coloring the clusters:

```
plot(z, col=km$cluster)
```



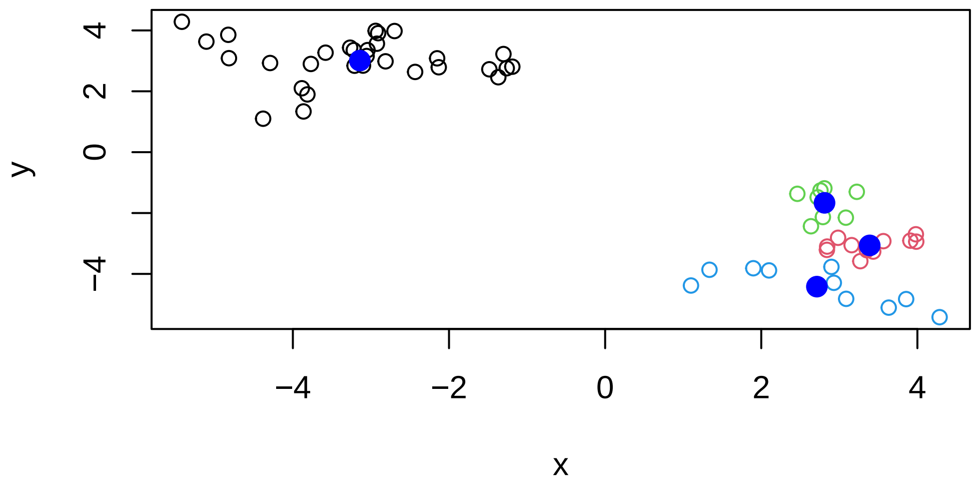
We can use the `points()` function to add new points to an existing plot like for te cluster centers:

```
plot(z, col=km$cluster)
points(km$centers, col="blue", pch=16, cex=1.5)
```



Q. Run km again and ask for 4 clusters and plot them

```
n <- 30
x <- (c(rnorm(n, -3), rnorm(n, +3)))
y <- rev(x)

z<- cbind(x,y)
km4 <- kmeans(z, centers = 4)
plot(z, col = km4$cluster)
points(km4$centers, col="blue", pch=16, cex=1.5)
```



Hierarchical Clustering

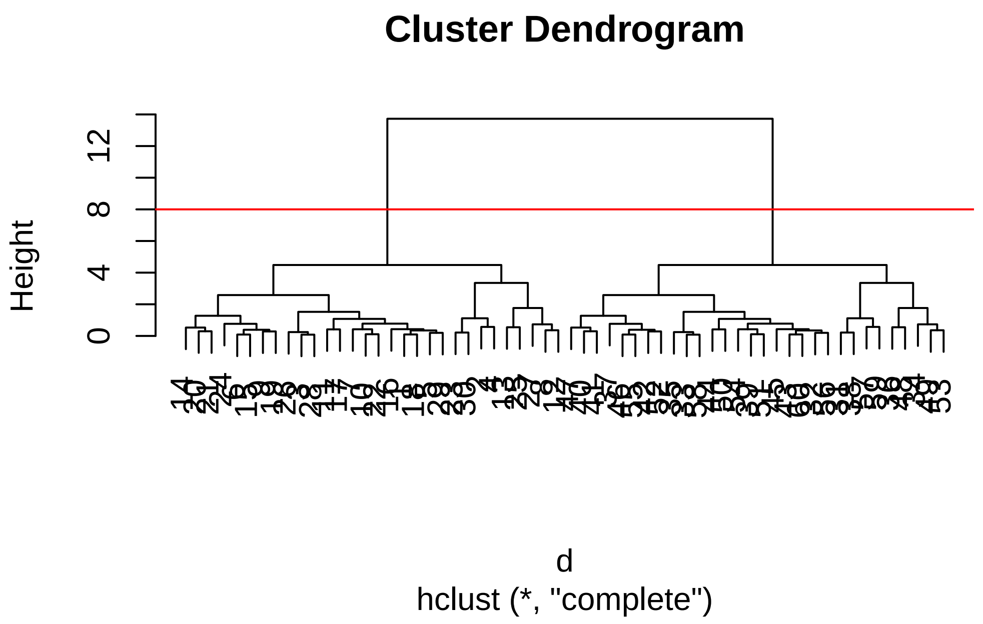
Let's take our same made-up data `z` and see how hclust works.

First we make a distance matrix of our data to be clustered:

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

```

Call:
hclust(d = d)

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

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



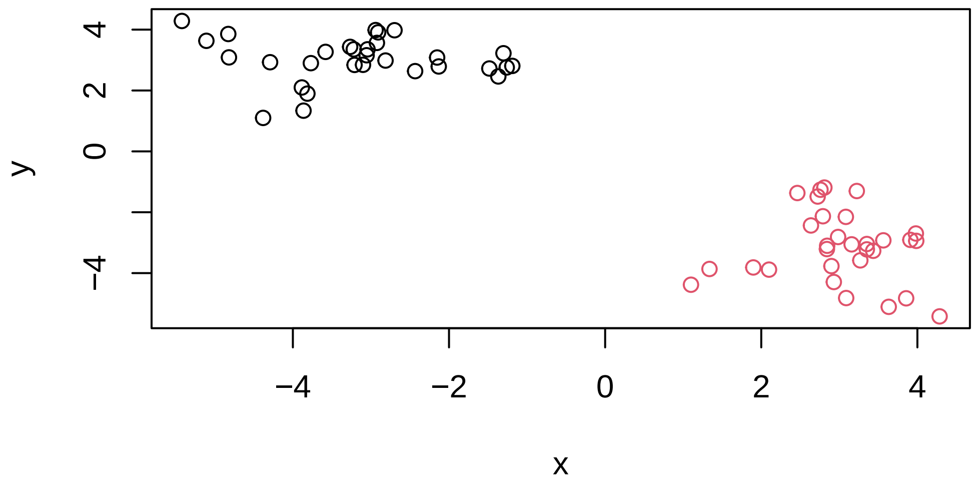
I can get my cluster membership vector by "cutting the tree" with the `cutree()` function:

```r
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 2 2
```

Can you plot `z` colored by our hclust results:

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



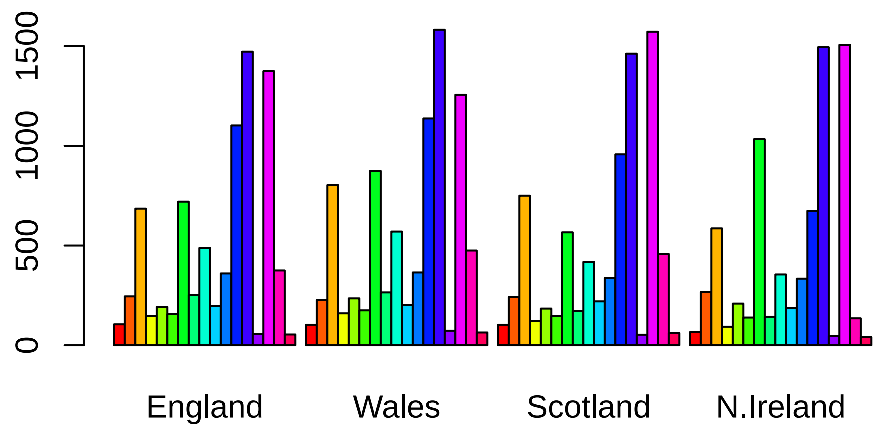
PCA of UK Food Data

Read data from the UK on food consumption in different areas

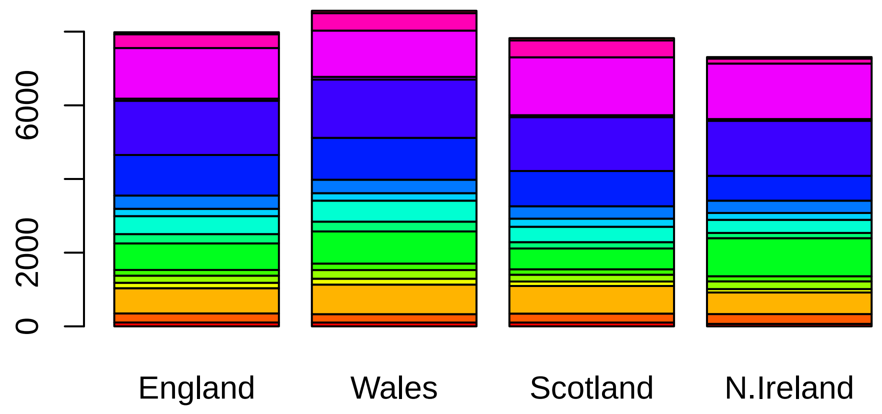
```r
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names = 1)
head(x)
```

```
               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
```

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

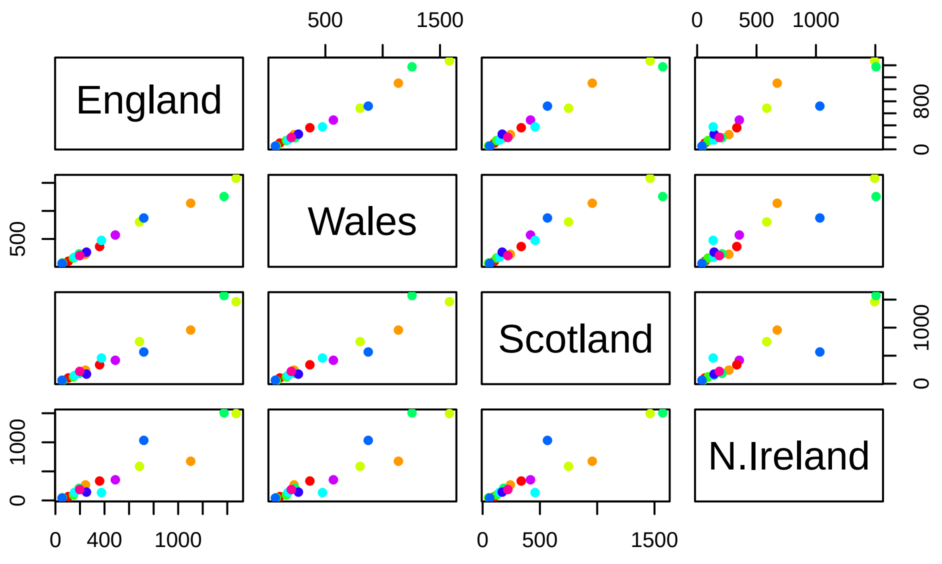


```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



A so-called pair-wise plot may help to compare countries and categories

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



It is difficult to see structure and trends in this small dataset- how can we compare when we have even larger data?! PCA to the rescue!

PCA

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

```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
```

Let's look at the `pca` object that we created from running `prcomp()`

```
attributes(pca)
```

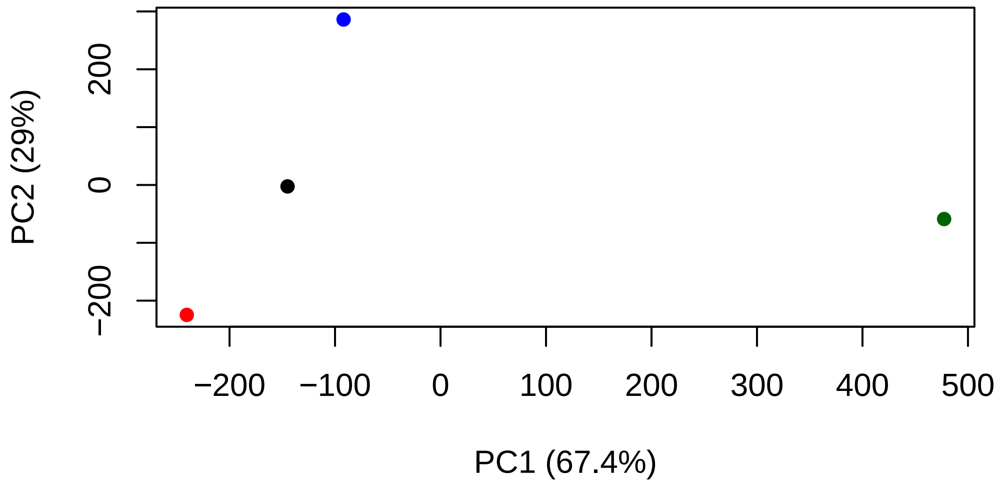
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```
pca$x
```

```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -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
```

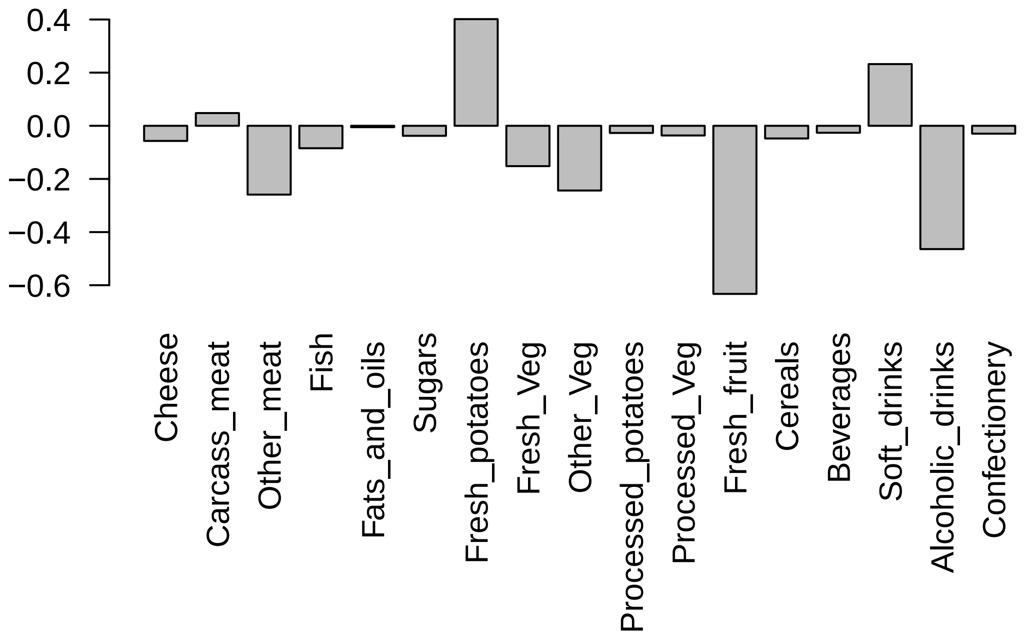
The PCA plot:

```
plot(pca$x[,1], pca$x[,2],
     col=c("black", "red", "blue", "darkgreen"), pch=16,
     xlab="PC1 (67.4%)", ylab="PC2 (29%)")
```



Bar plot:

```r
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```


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