Class07

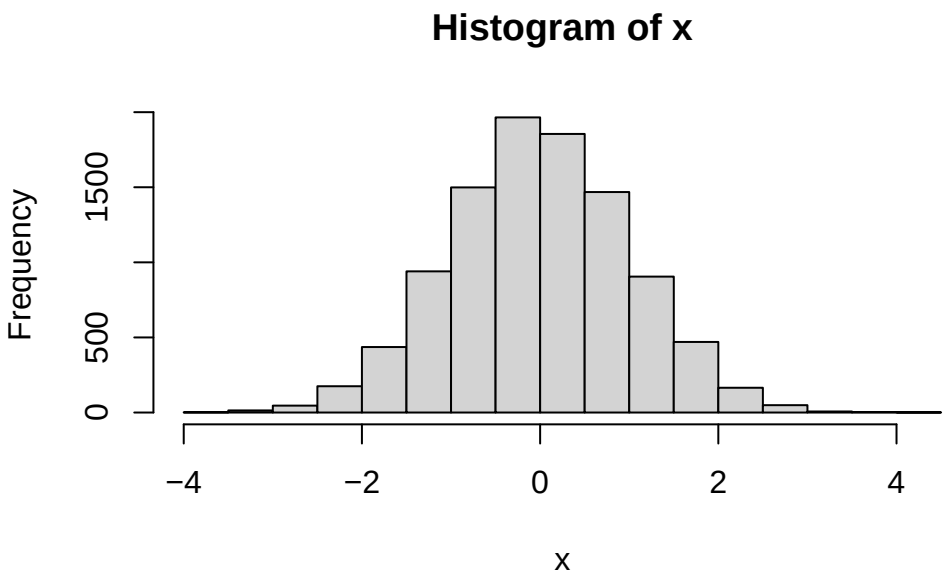
Natalie Ogg, A91030809

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 what the answer should be.

```
x <- rnorm(10000)
hist(x)
```



`rnorm()` returns random values

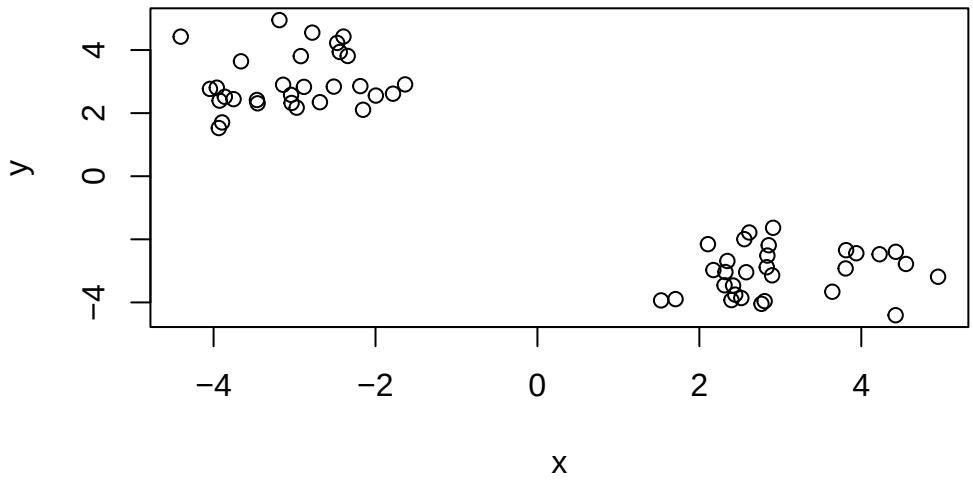
60 points

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

```
           x         y
[1,] 2.442494 -3.752964
[2,] 2.105978 -2.153749
[3,] 1.527616 -3.935536
[4,] 4.424140 -4.405746
[5,] 3.806614 -2.922511
[6,] 2.910423 -1.634642
```

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

```
plot(x)
```



Now let's see if kmean recognizes the 2 clusters we see above.

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

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

Cluster means:
          x         y
1  2.990097 -3.031402
2 -3.031402  2.990097

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 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] 39.24981 39.24981
 (between_SS / total_SS =  93.3 %)

Available components:

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

Q1. How many points are in each cluster?

```
k$size
```

```
[1] 30 30
```

Q2. Cluster membership?

```
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 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
```

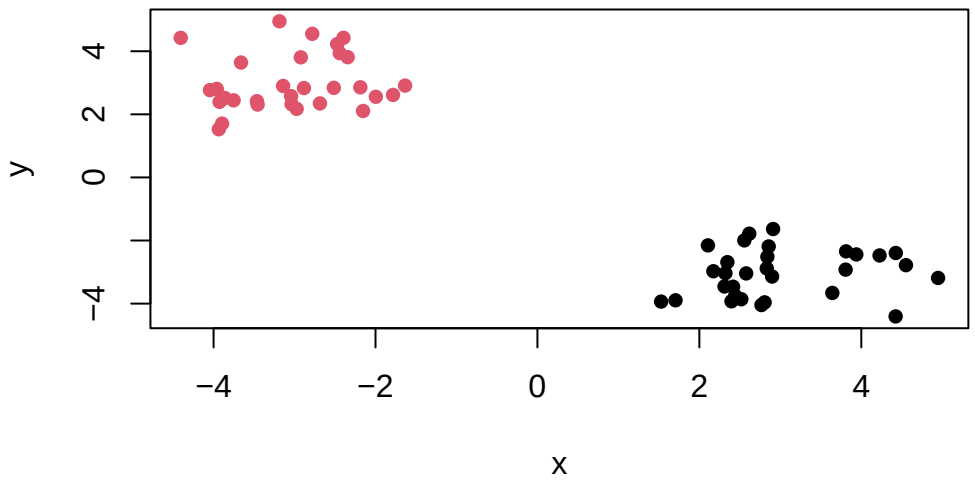
Q3. Cluster centers?

```
k$centers
```

```
          x         y
1  2.990097 -3.031402
2 -3.031402  2.990097
```

Q4. Plot my clustering results

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



Q5. Cluster the data again into 4 groups and plot the results.

```
k4 <- kmeans(x, centers=4)
k4
```

```
K-means clustering with 4 clusters of sizes 8, 9, 13, 30

Cluster means:
          x         y
1  2.632960 -2.230037
2  4.197337 -2.956857
3  2.374091 -3.576157
4 -3.031402  2.990097
```

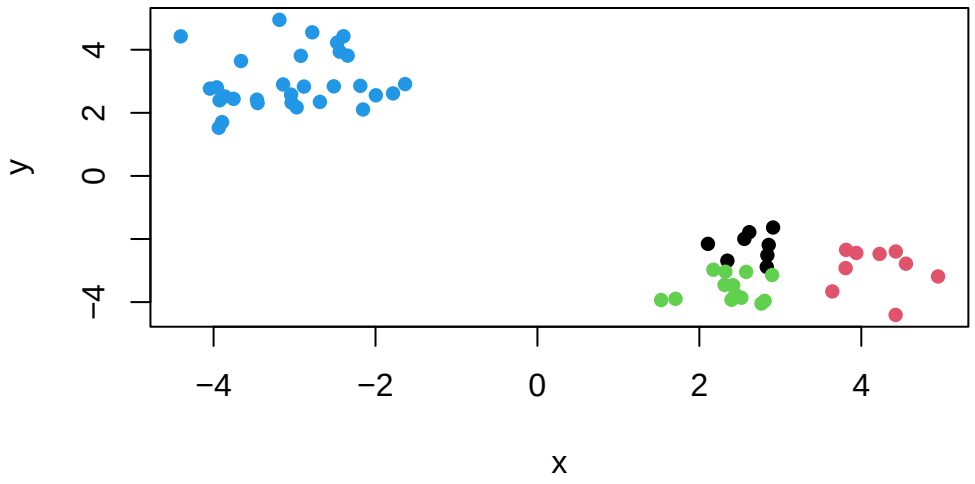
```
Clustering vector:
 [1] 3 1 3 2 2 1 1 2 1 1 2 3 2 1 3 3 2 3 1 3 2 3 2 3 2 3 1 3 3 3 4 4 4 4 4 4 4 4
[39] 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4

Within cluster sum of squares by cluster:
[1]  1.909133  5.341262  3.883817 39.249806
 (between_SS / total_SS =  95.7 %)

Available components:

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

```r
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 that you need to tell it how many groups (k, or centers) you want.

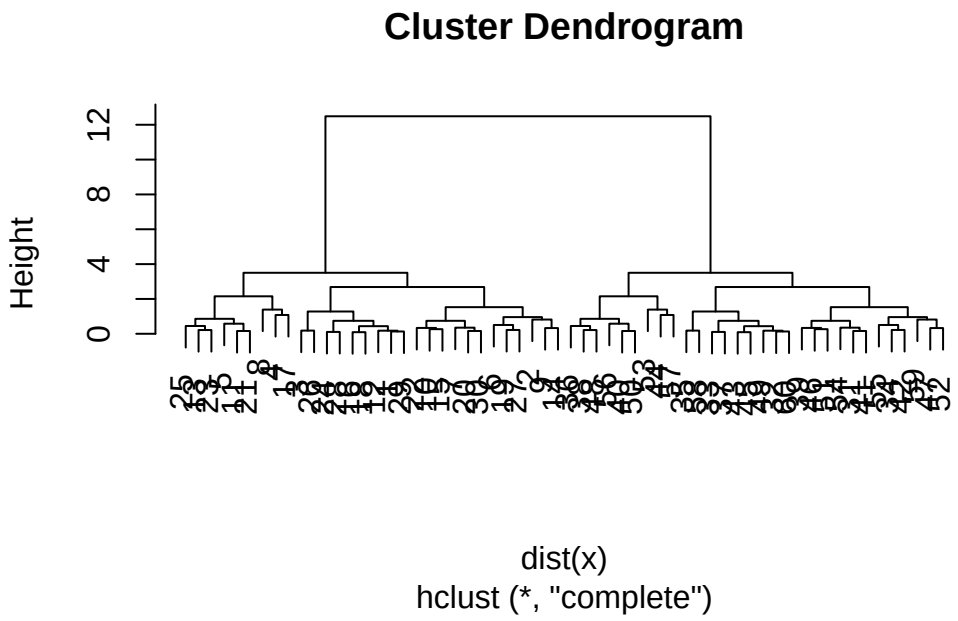
Hierarchical clustering

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

You generate this matrix with the `dist()` function

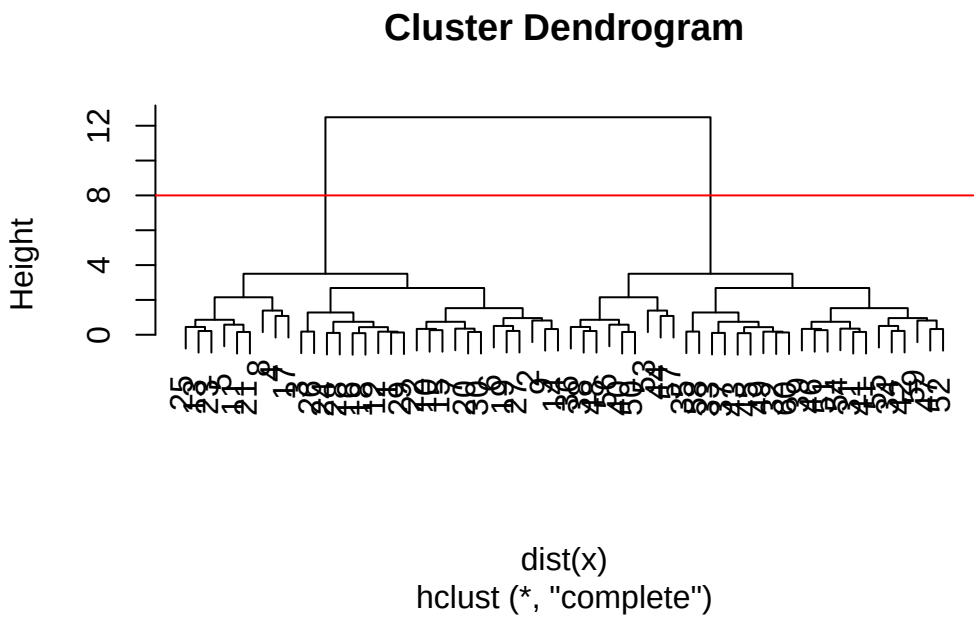
```
hc <- hclust(dist(x))
```

```
plot(hc)
```



To find clusters (cluster membership vector) from an `hclust()` result, we can cut the tree at a certain height that we like using the `cutree()` function.

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



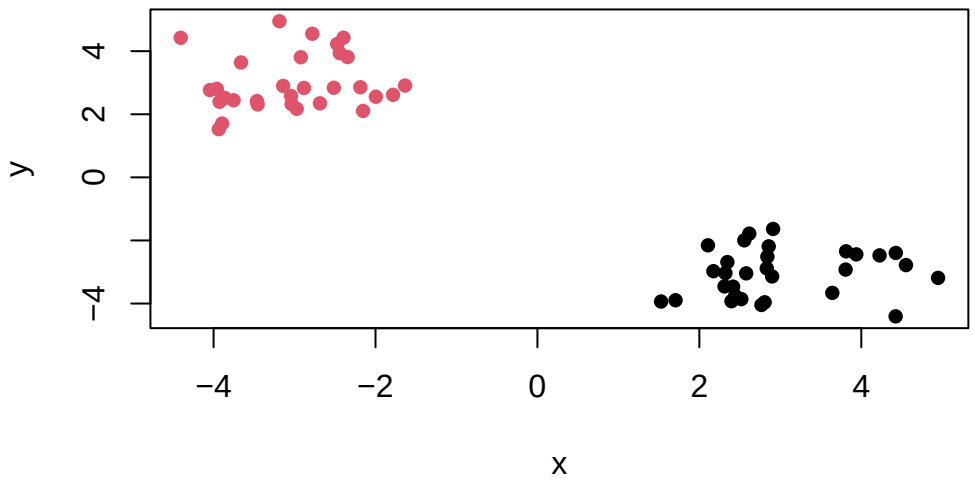

```
grps <- cutree(hc, h=8)
```

```
table(grps)
```

```
grps
 1  2
30 30
```

Q6. Plot our hclust results

```
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 in 1997.

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
```

Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

Complete the following code to find out how many rows and columns are in x?

```
dim(x)
```

```
[1] 17  5
```

A: 17 rows, 5 columns

Preview the first 6 rows

```
head(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
```

We need to assign the character values in column1 (x) to be the rownames, then delete column1 so only the rownames are left where they should be.

```
row.names(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
```

```
dim(x)
```

```
[1] 17  4
```

Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

The approach I used is risky because if you enter the wrong values you could erase data on accident.

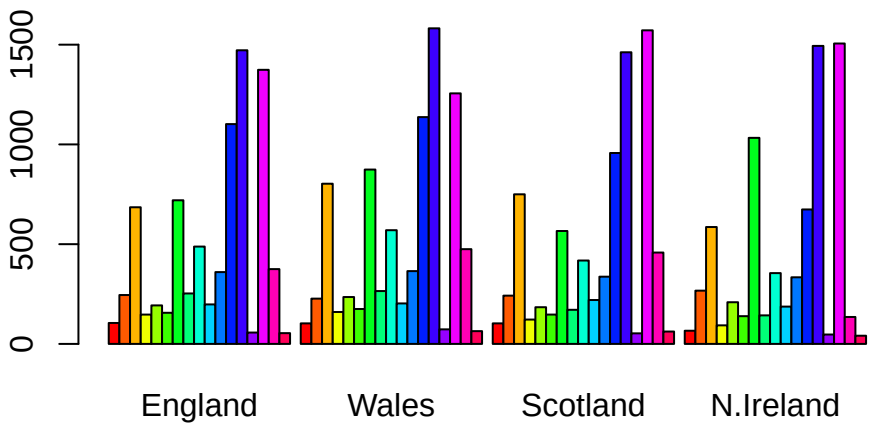
The better way to do it would be assigning x to row names right off the bat:

```
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
```

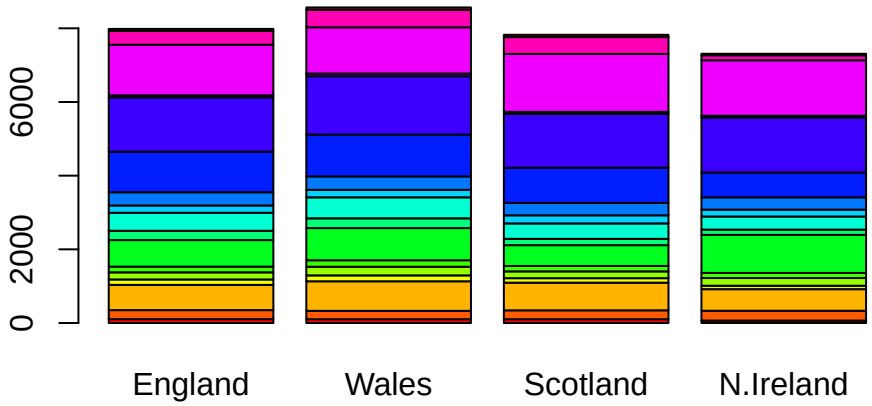
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



Q3: Changing what optional argument in the above barplot() function results in the following plot?

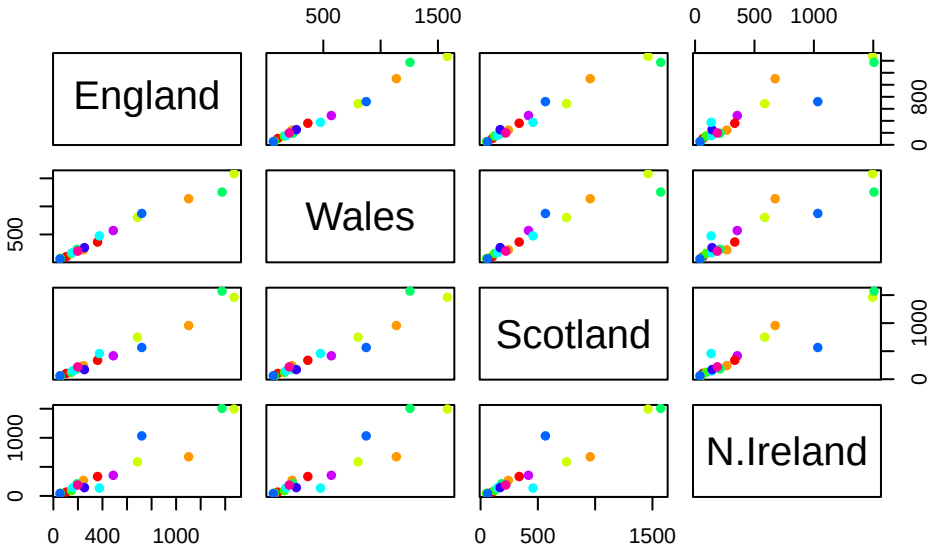
By setting `beside = FALSE`

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



Q5: Generating all pairwise plots may help somewhat. Can you make sense of the following code and resulting figure? What does it mean if a given point lies on the diagonal for a given plot?

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



Each plot is a comparison of two countries. If a dot lies on the diagonal, then the two countries being compared have equal consumption for the food represented by that point. If a dot is further to one side of the diagonal, the food it represents is more conumed by the country on that side. The closer points are to a perfect diagonal for a given plot, the more similar the consumption is in those two countries.

PCA - Principal Component Analysis

PCA can help us make sense of these datasets.

The main function in base R is `prcomp()`

```
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
```

We want switch the variables, so the countries are the rows and the foods are the columns. We transpose this dataset using function `t()`.

```
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 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
```

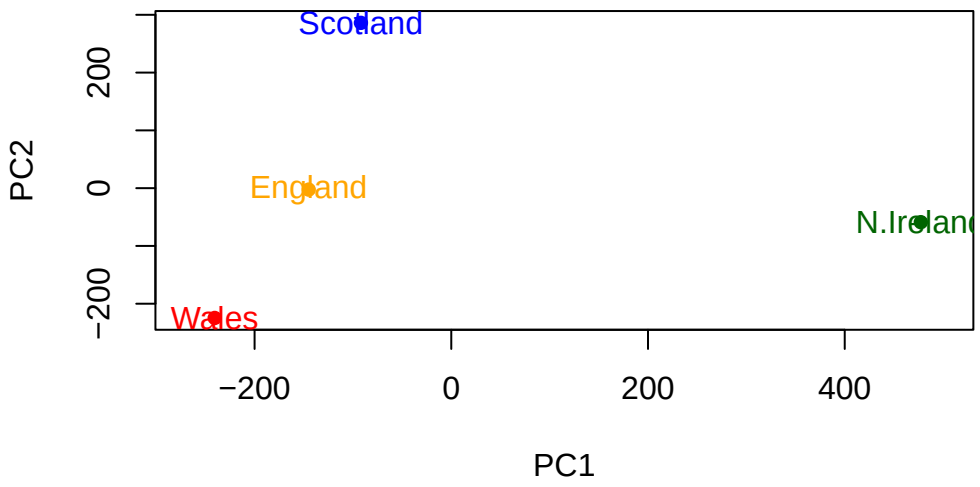
Proportion of variance in each PCA captures more of the variance, and it accumulates as you go along from PC1 to PC3 being 1.00000

```
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
```

You don't need all the data, you can just compare these PCAs and their variance.

```
plot( pca$x[,1], pca$x[,2], col=c("orange", "red", "blue", "darkgreen"), pch=16, xlab="PC1
text(pca$x[,1], pca$x[,2], colnames(x), col=c("orange", "red", "blue", "darkgreen"))
```




```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

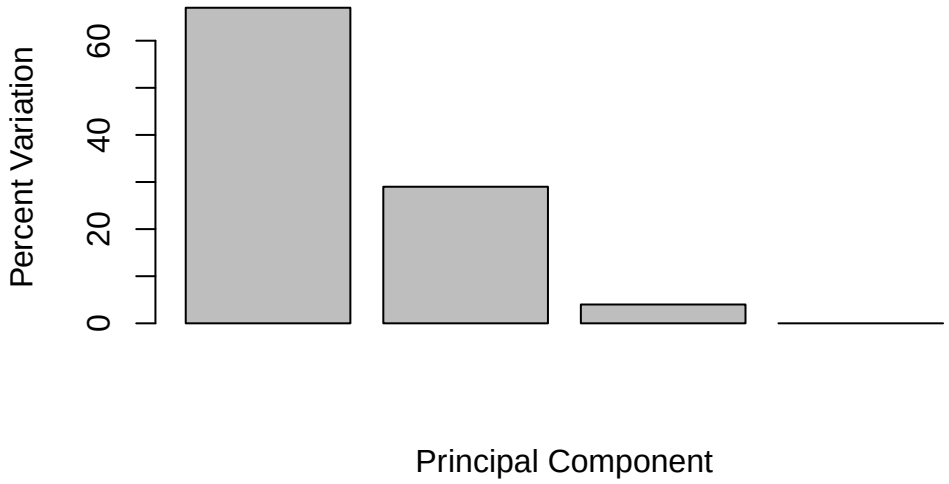
```
[1] 67 29  4  0
```

```
z <- summary(pca)
z$importance
```

```
                            PC1       PC2      PC3           PC4
Standard deviation     324.15019 212.74780 73.87622 2.921348e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

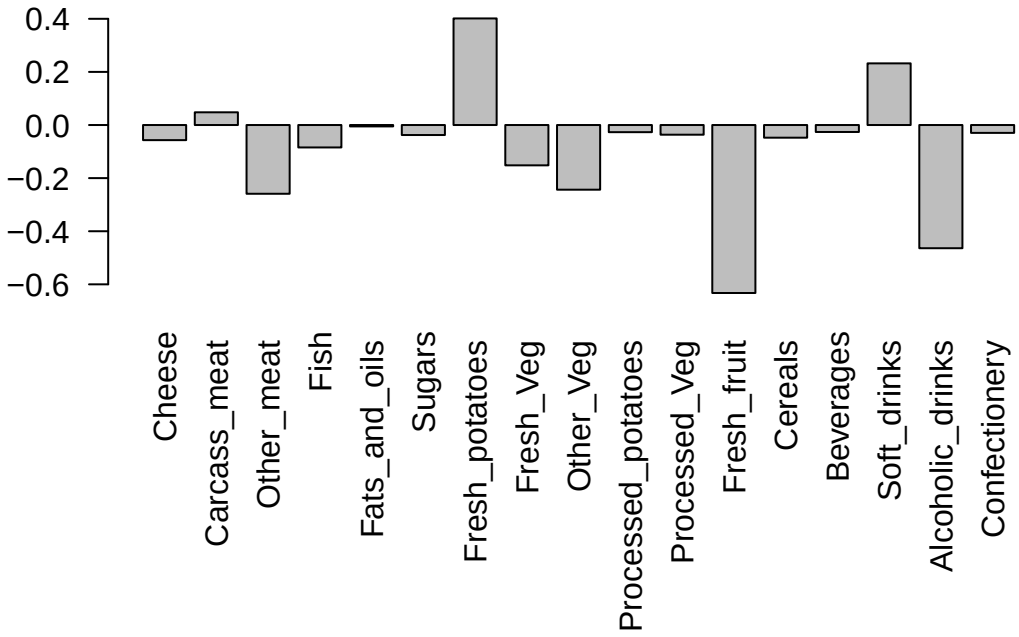
Here we summarize the variances in a bar plot

```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



Then we 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