Class 07 Unsupervised learning

Andre Modolo

In this class we will explore clustering and dimensional reduction methods.

K-means (K=number of clusters)

Make up input data where we know what the answer should be.

```
tmp <- c(rnorm(30, -3), rnorm(30, 3))
# make it into a 2 dimensional thing
x <- cbind(x=tmp, y=rev(tmp))
# rev(tmp) flips the vector
rev(tmp)
```

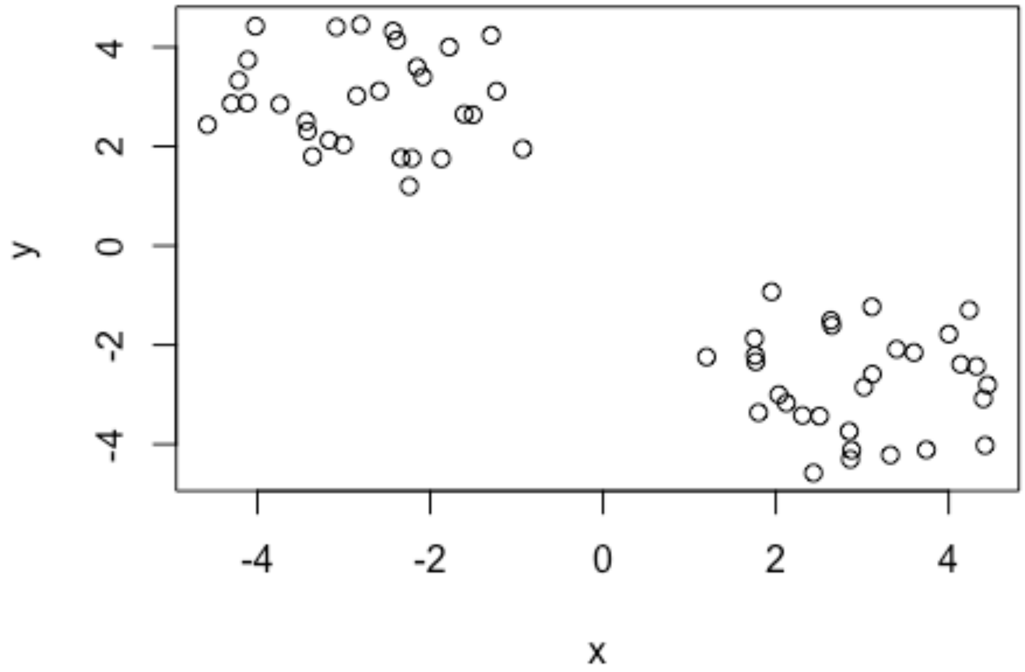
```
 [1]  4.0016191  2.6483198  2.5064679  3.3253771  3.1117622  3.5980262
 [7]  2.0347113  4.4019919  2.8492427  3.1169560  1.7607587  4.3224521
[13]  2.6365857  2.8629020  2.4357906  2.8770494  4.4513608  4.2381046
[19]  1.7683701  3.3986070  3.0199585  1.8001638  2.3082575  3.7419589
[25]  1.7525084  2.1219547  4.1407419  1.1984598  4.4219687  1.9498403
[31] -0.9288244 -4.0223417 -2.2423369 -2.3878829 -3.1700741 -1.8698086
[37] -4.1114540 -3.4197867 -3.3617979 -2.8492117 -2.0849020 -2.3381561
[43] -1.2947142 -2.8047979 -4.1171737 -4.5780154 -4.3020050 -1.5038790
[49] -2.4295975 -2.2111232 -2.5886221 -3.7385883 -3.0853108 -3.0020882
[55] -2.1526184 -1.2318432 -4.2169147 -3.4346825 -1.6078447 -1.7804223
```

```
head(x)
```

```
             x        y
[1,] -1.780422 4.001619
[2,] -1.607845 2.648320
[3,] -3.434683 2.506468
[4,] -4.216915 3.325377
[5,] -1.231843 3.111762
[6,] -2.152618 3.598026
```

Quick plot of x to see the 2 groups around (-3, 3) and (3, -3)

```
plot(x)
```



Use the kmeans() function setting k to 2 and nstart=20 (do the picking points and finding the distances to decide a potential cluster 20 times before deciding a winning set of clusters)

```
km <- kmeans(x,center=2, nstart=20 )
km

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

Cluster means:
          x         y
1 -2.762227  2.960076
2  2.960076 -2.762227

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] 56.24624 56.24624
 (between_SS / total_SS =  89.7 %)
```

```
Available components:

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

Clustering means: gives us the mean point of each cluster

```
km$size

[1] 30 30
```

Size of the clusters found: 30 and 30 Clustering vector: lables each component of the vector as the first or second cluster

Q. What component of your result object details? - cluster assignment/membership (1 or 2 in this case)

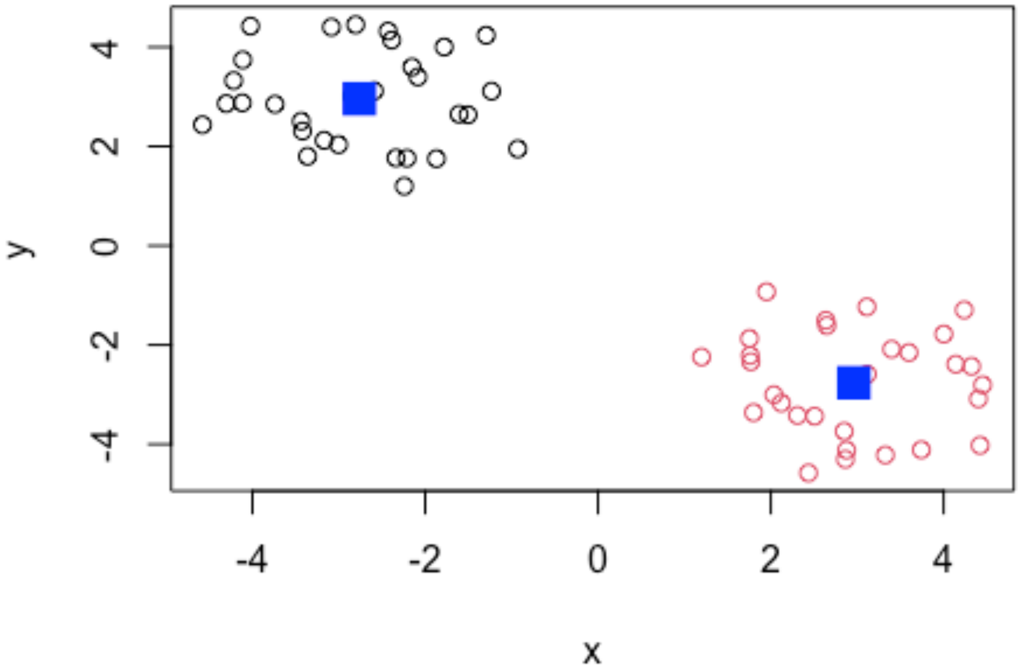
```
km$cluster

 [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$center

          x         y
1 -2.762227  2.960076
2  2.960076 -2.762227
```

Q. plot x colored by the kmeans cluster assignment and add cluster centers as blue points

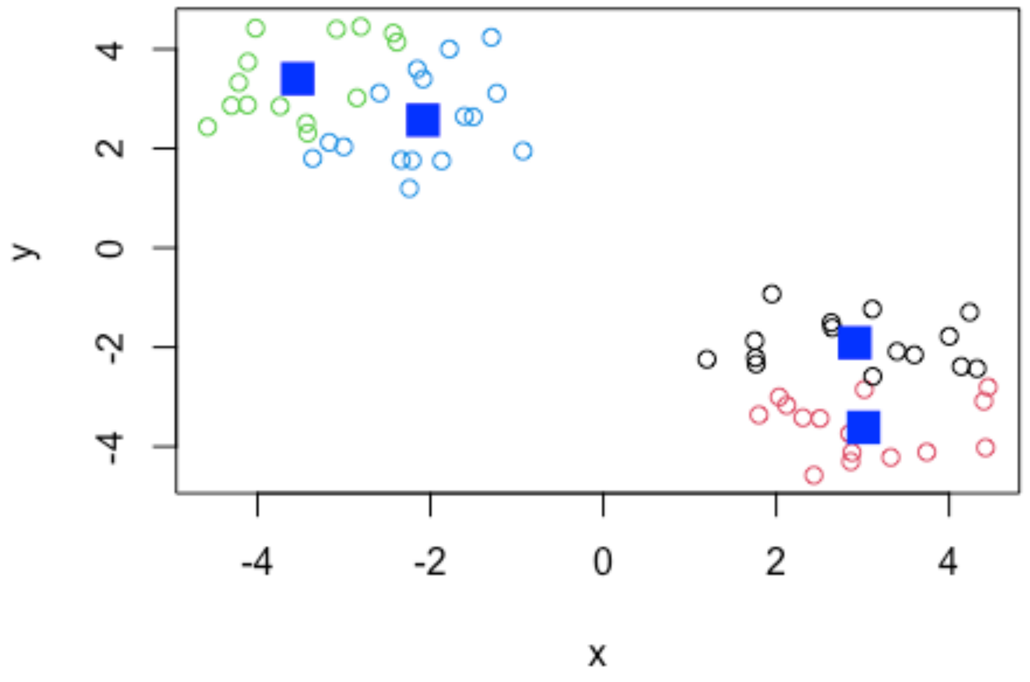
```
plot(x, col=km$cluster)
points(km$centers, col="blue", pch=15, cex=2)
```



What if I ask for more than 2 clusters?

```
km4 <- kmeans(x, 4, nstart=20)
km4

K-means clustering with 4 clusters of sizes 15, 15, 14, 16

Cluster means:
          x         y
1  2.909541 -1.910172
2  3.010610 -3.614283
3 -3.535554  3.404680
4 -2.085566  2.571047

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

Within cluster sum of squares by cluster:
[1] 18.72786 15.66181 15.46351 19.89540
 (between_SS / total_SS =  93.6 %)
```

```
Available components:

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

plot(x, col=km4$cluster)
points(km4$centers, col="blue", pch=15, cex=2)
```



#Hierarchical Clustering

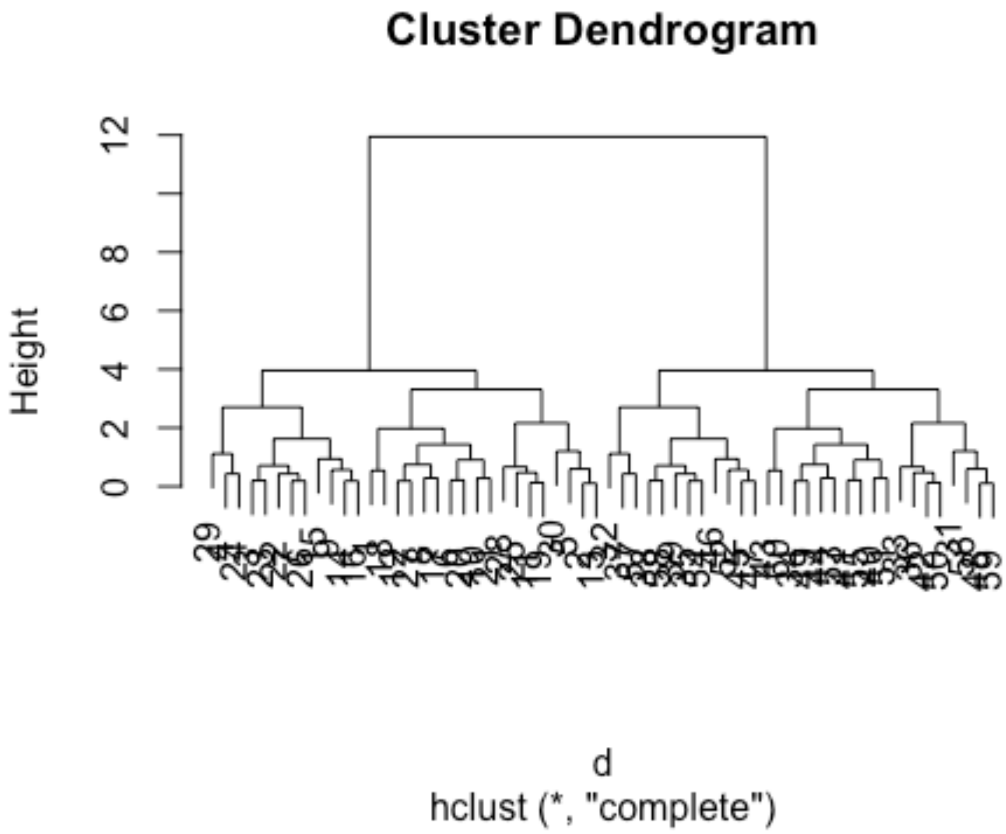
Super useful and widely employed clustering method which has the advantage over kmeans because it can show you a little something about the true nature of the clustering in your data You need to give it a “d” distance matrix as an input (how far apart the values are). get it using `dist()`

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


Call:
hclust(d = d)
```

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

There is a plot method for hcluster results:

```
plot(hc)
```

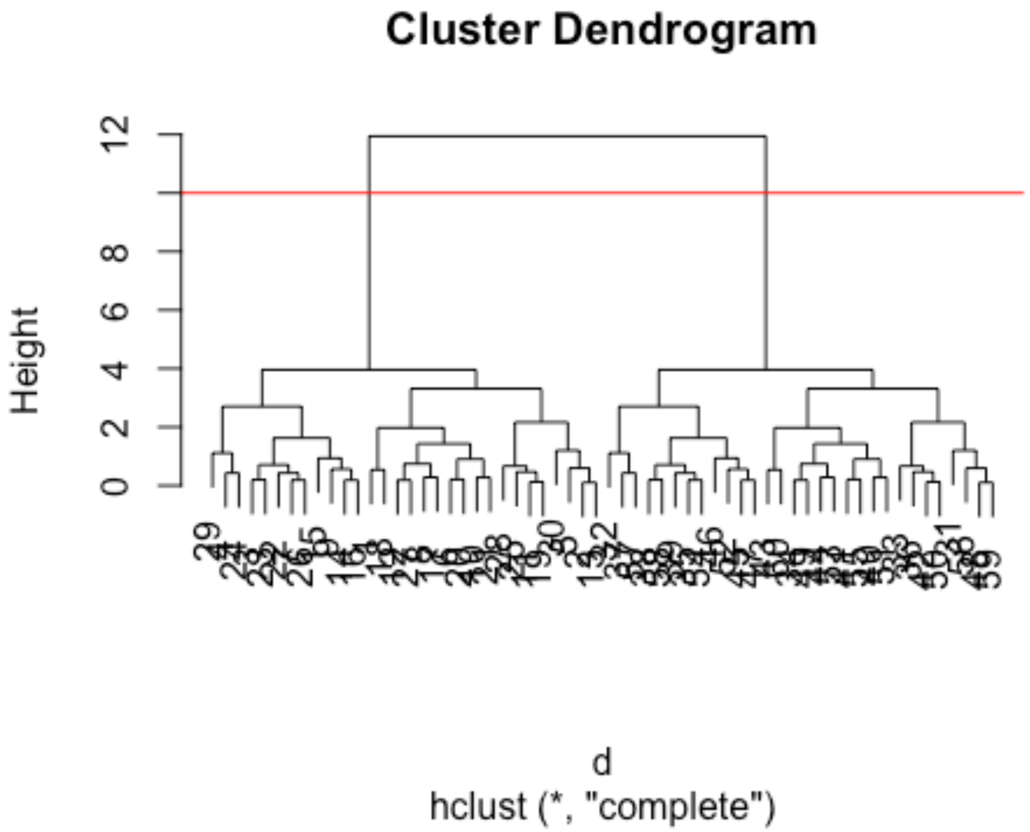


You get 2 overall branches with 1:30 on one branch and 31:60 on the other branch. This makes sense because in the vector we made the first 30 numbers have a set mean and the second 30 numbers have another set mean.

Long goal post = big jump between the things you grouped together and the next group.

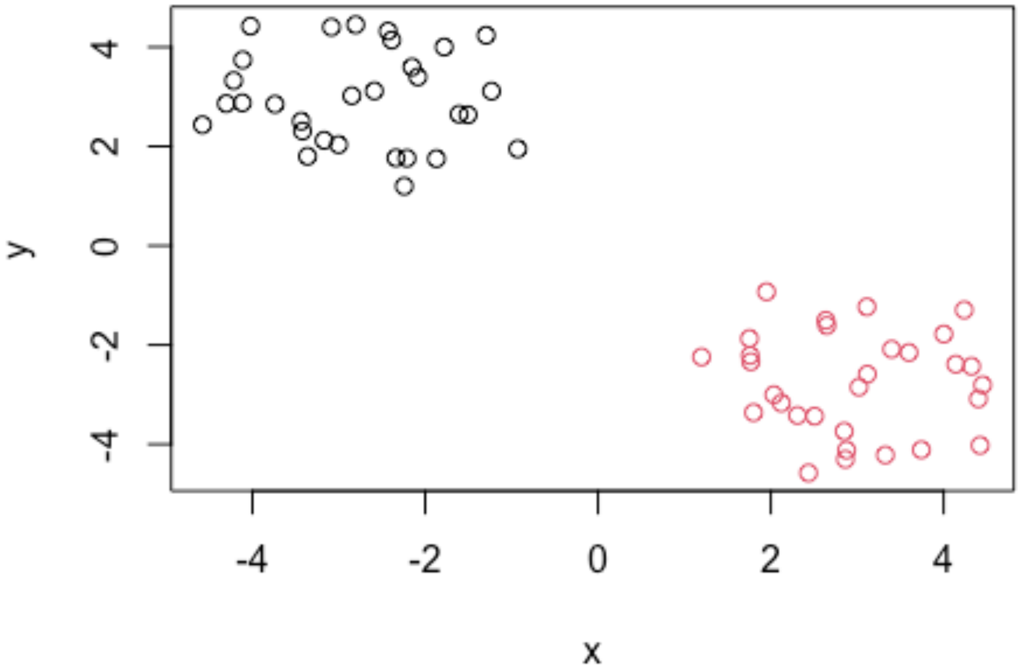
How do I get an actual result out of this? cut the longest post, and you are left with “subtrees” in this case you are left with 2 subtrees.

```
plot(hc)
#cut the tree with this line
abline(h=10, col="red")
```



To get the cluster membership vector, I need to "cut my tree" to yield subtrees with the function `cutree()` with h=height to cut, **or** k= number of clusters you want after the cut

```
grps <- cutree(hc, h=10)
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

plot(x, col=grps)
```



Principal Component Analysis (PCA)

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

Import the food data from the 4 countries

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
dim(x)

[1] 17  5
```

There are 17 row and 5 columns

```
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 can remove the x column and only get the 4 counties as columns by using this code

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

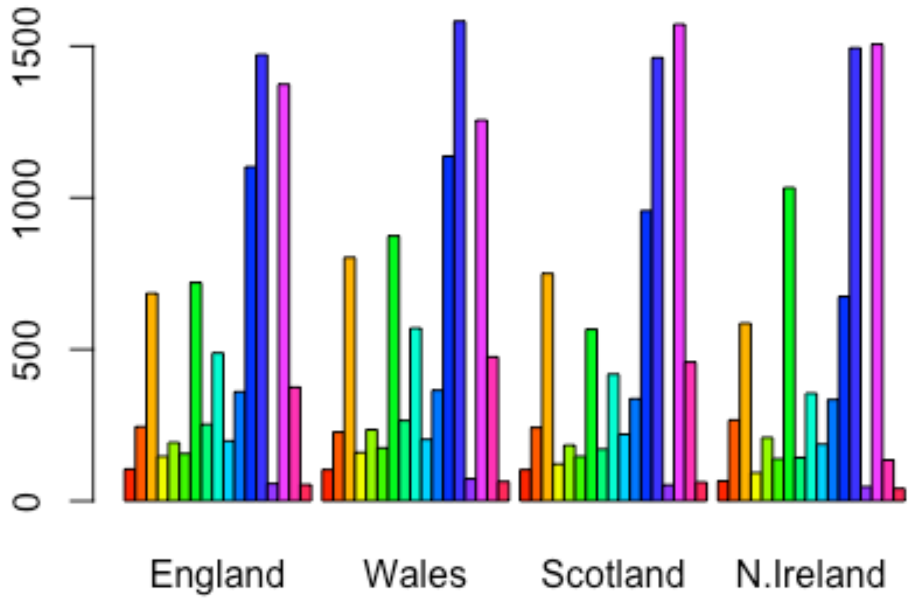
Be careful with this approach because if you keep running it multiple times, it will keep removing the name of the first column

A more robust way of doing it would be using this method, setting the row name as 1, so you can rerun the code and it won't delete any more column names

```
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
```

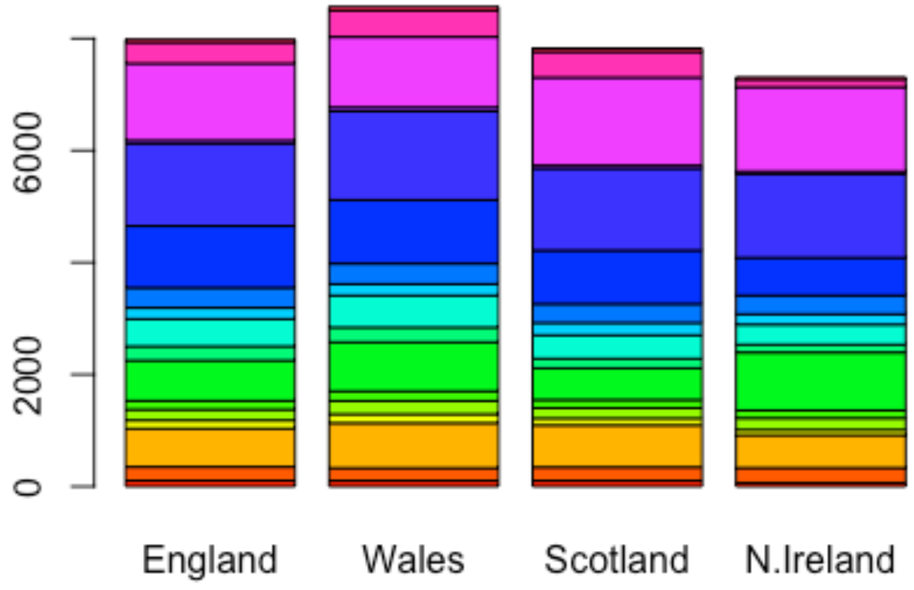
Spotting the major differences and trends using a bar plot

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



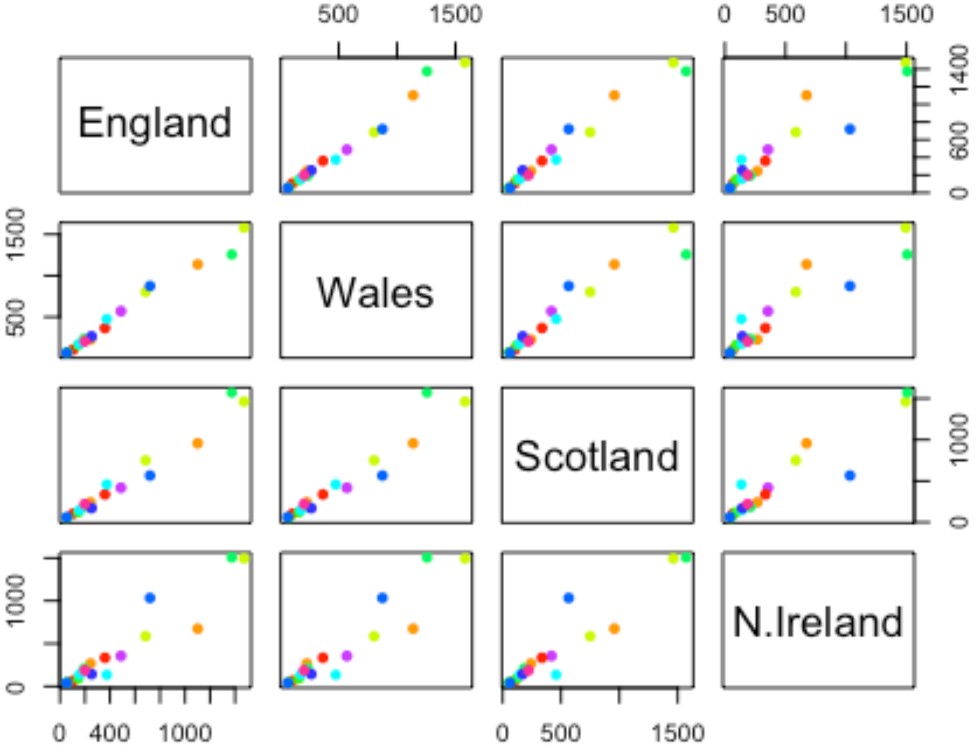
Doing `beside=False` you get this kind of bar plot

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



What about plotting it this way?

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



What does it mean when a point lies on a diagonal of a given plot? This gives a matrix of scatterplots comparing the countries as an x and a y variable in each situation. This way you only have to look at bottom left or top right half depending in which country you want to be on which axis.

If the point lies on the diagonal of a scatterplot, this means that the two countries have a similar amount of consumption for that specific food group (color)

The main difference in food consumption between N. Ireland and the other countries is in the food colored blue

#PCA to the rescue

Take the transpose of x to flip the rows and columns

```
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
```

Now do prcomp() and print out the summary

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

Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 4.189e-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: 67.4% of all the variance is captured on the new axis made.

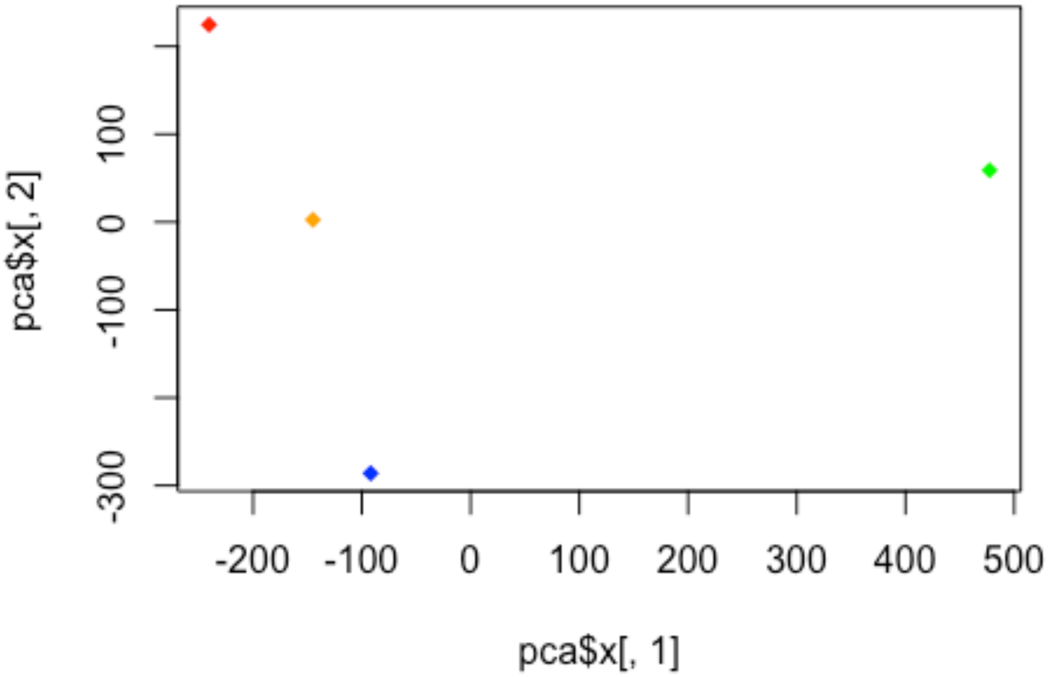
Cumulative Proportion: adding 2 or 3 PCs together you capture basically all the variance from the plots (ex: PC2 with 96.5%!)

A “PCA plot” (a.k.a “Score Plot”, PC1vsPC2 plot, etc.)

```
pca$x

                 PC1         PC2         PC3           PC4
England   -144.99315    2.532999 -105.768945  2.842865e-14
Wales     -240.52915  224.646925   56.475555  7.804382e-13
Scotland   -91.86934 -286.081786   44.415495 -9.614462e-13
N.Ireland  477.39164   58.901862    4.877895  1.448078e-13
```

Plot the PC1 vs PC2 and color the countries Irland is green

```
plot(pca$x[,1], pca$x[,2], col=c("orange","red", "blue", "green"), pch=18)
```



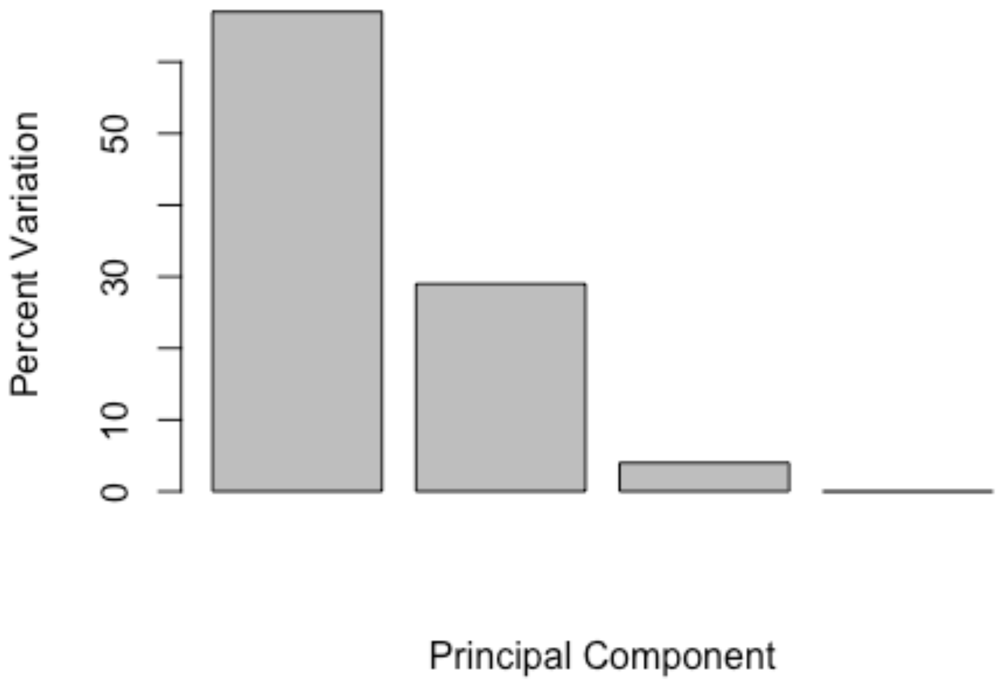
You see that N. Ireland is actually different than the other countries in their food consumption.

Below we can use the square of pca$sdev , which stands for "standard deviation", to calculate how much variation in the original data each PC accounts for:

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

[1] 67 29  4  0
```

This information can be summarized in a plot of the variances (eigenvalues) with respect to the principal component number (eigenvector number), which is given below.

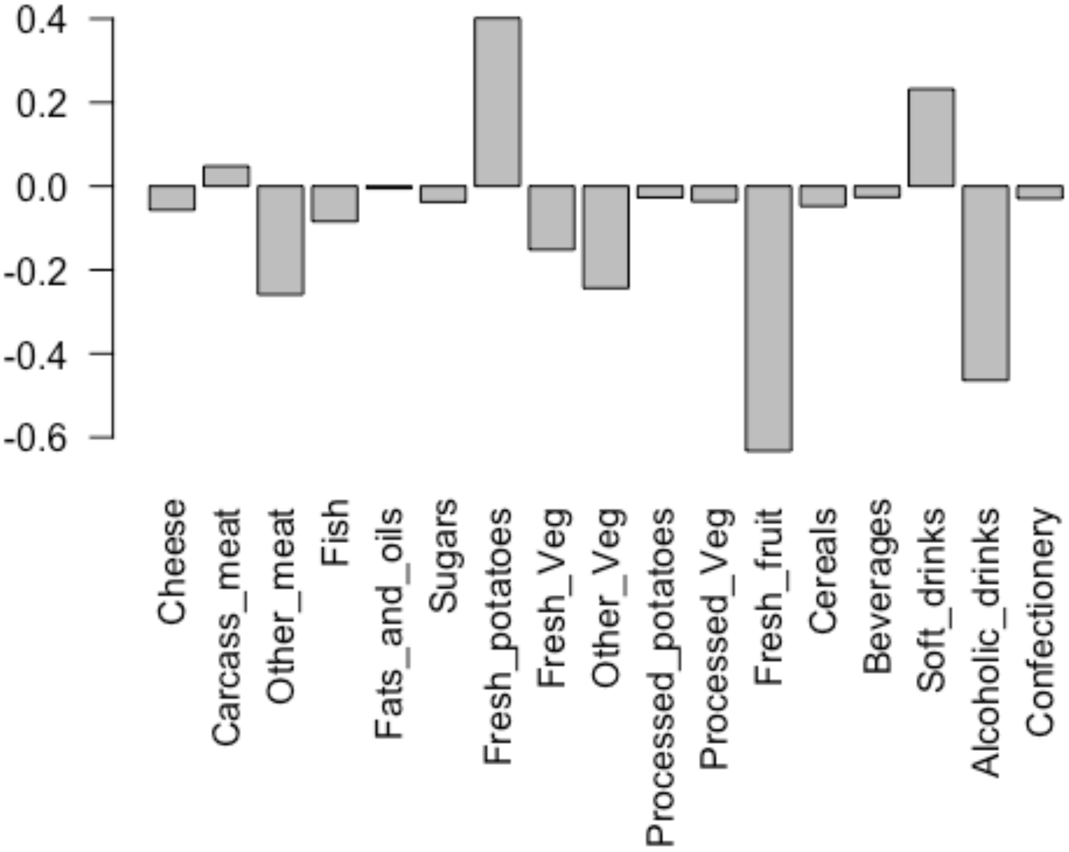
```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



We can also consider the influence of each of the original variables upon the principal components (typically known as loading scores). This information can be obtained from the prcomp() returned $rotation component

Using PC1 we can get this barplot:

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



Now we can see what foods that make N. Ireland more different than the rest if the countries.