Class 07: Machine Learning 1

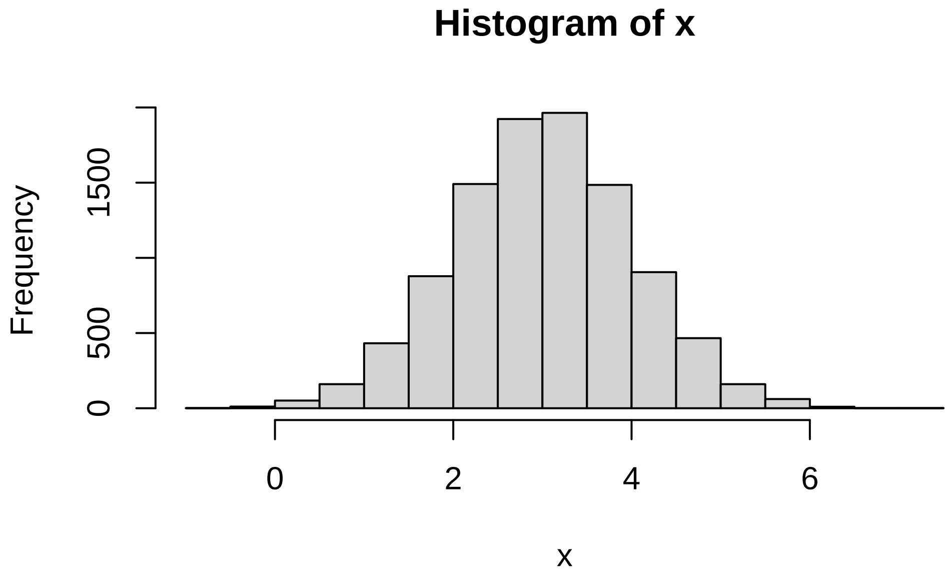
Eli Haddad (A16308227)

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, mean =3)
hist(x)
```

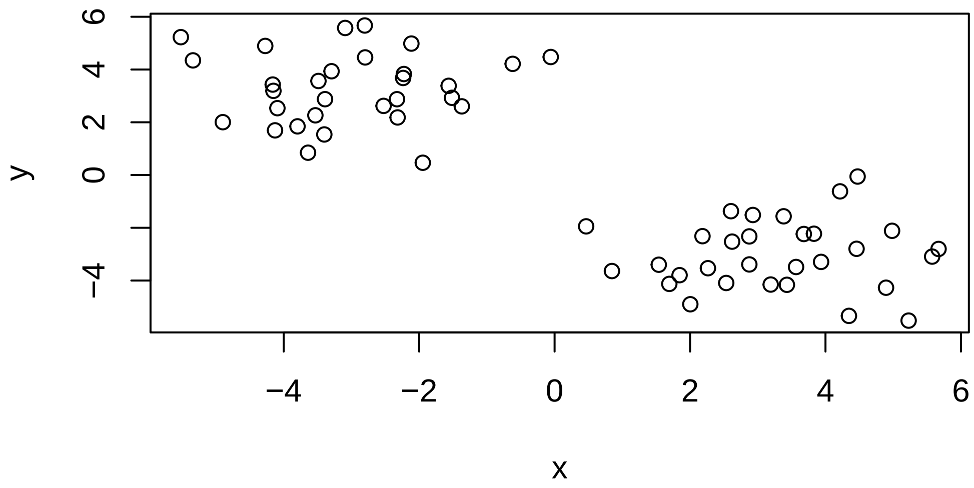


60 points

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

```
            x           y
[1,] 4.474437 -0.05718224
[2,] 2.874877 -2.32710569
[3,] 4.894207 -4.27255986
[4,] 4.214492 -0.61889480
[5,] 2.604095 -1.37036626
[6,] 2.264292 -3.53226741
```

```
plot(x)
```



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

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

```
Cluster means:
          x         y
1  3.271259 -3.020567
2 -3.020567  3.271259

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] 103.0179 103.0179
 (between_SS / total_SS =  85.2 %)

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 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
```

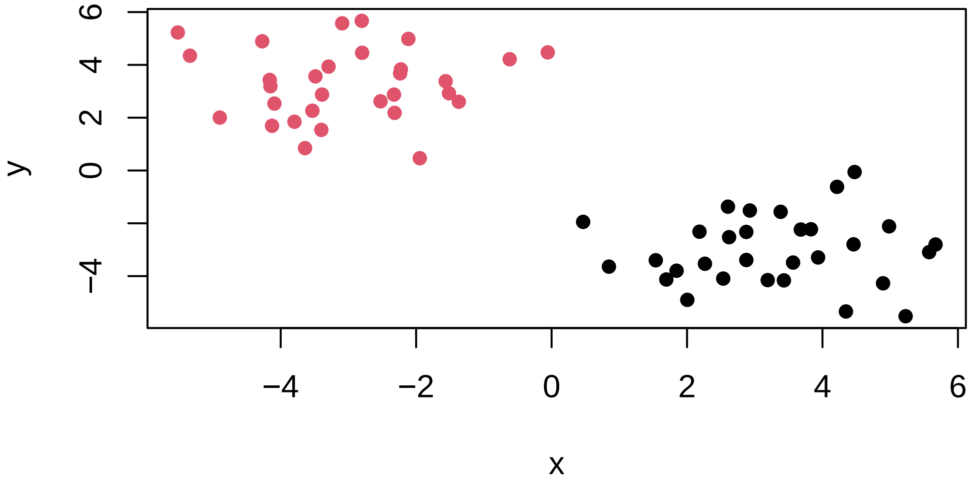
Q3. Cluster centers?

```
k$centers
```

```
          x         y
1  3.271259 -3.020567
2 -3.020567  3.271259
```

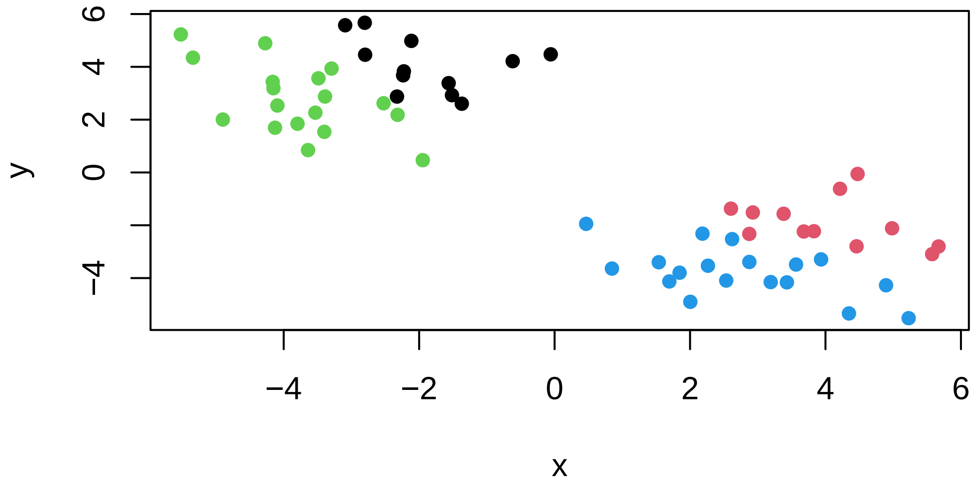
Q4. Plot my clustering results

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



Q5. Cluster the data again with kmeans() into 4 groups and plot the results.

```r
k4 <- kmeans(x, centers= 4, nstart = 20)
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 in that you need to tell it how many groups (k, or centers) you want.

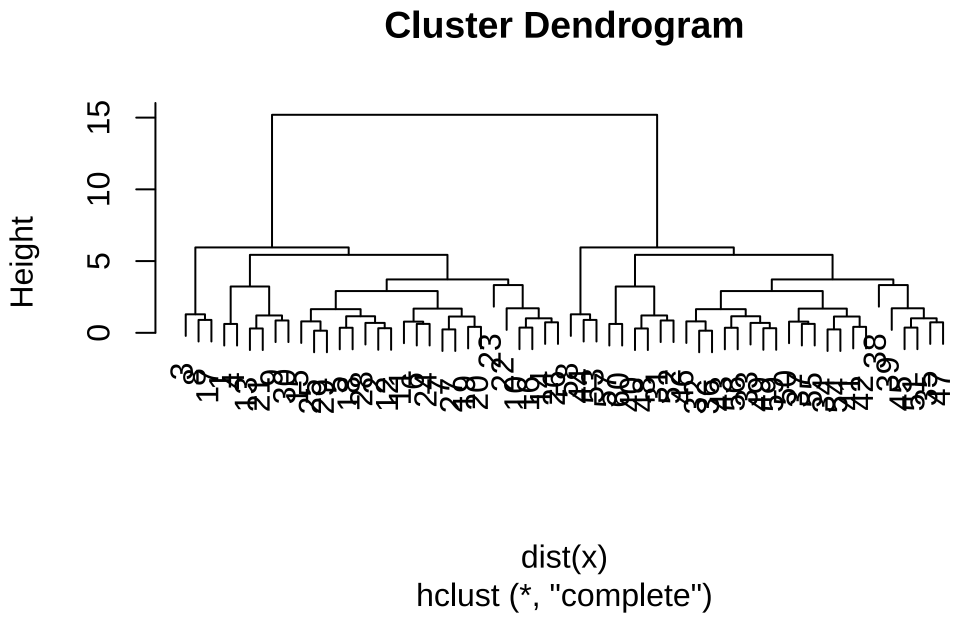
Hierarchial Clustering

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

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

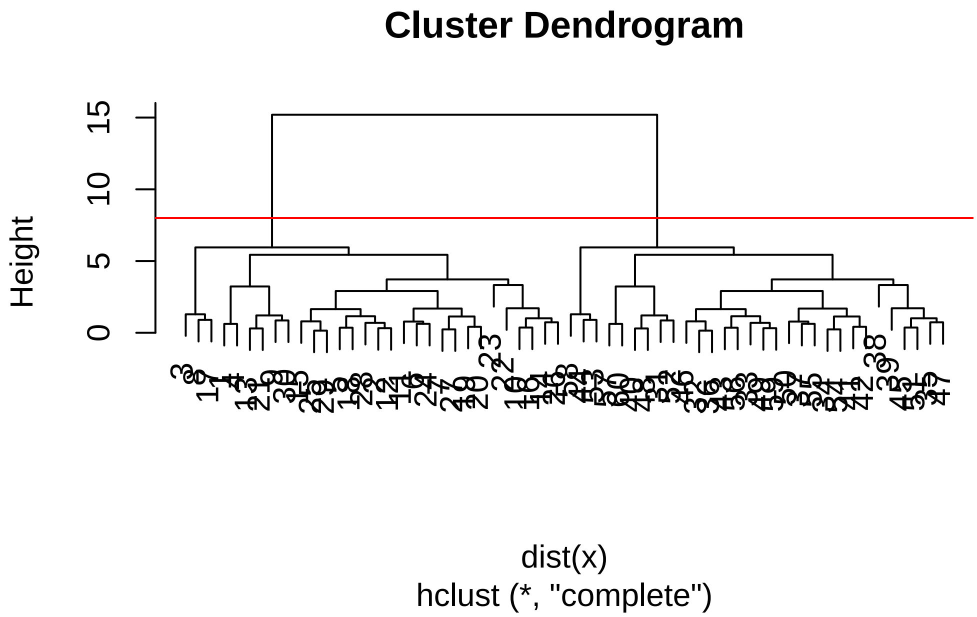
```
Call:
hclust(d = dist(x))

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

```
plot(hc)
```



To find the clusters (cluster membership vector) from `hclust()` result we can “cut” the tree at a certain height that we like. For this we can use the `cutree()` function.

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



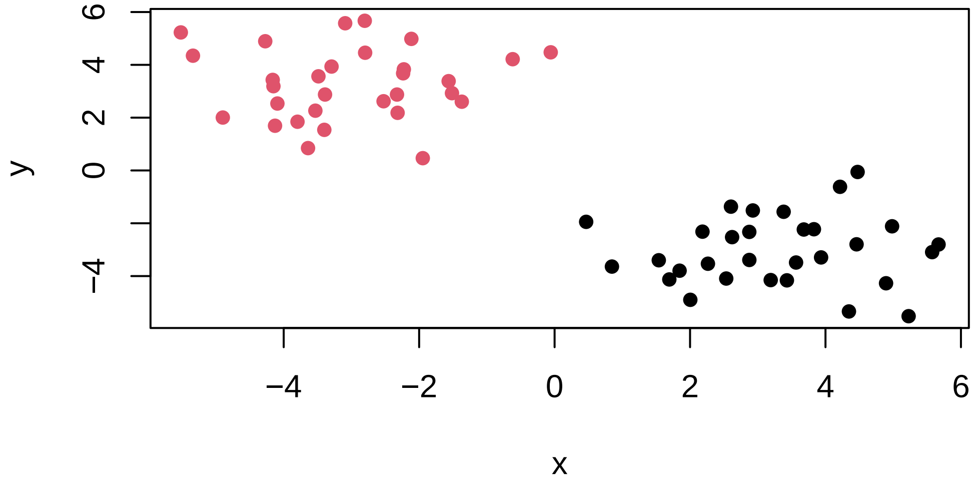
```r
grps <- cutree(hc, h=8)
```

```r
table(grps)
```

```
grps
 1  2 
30 30 
```

Q6. Plot our hclust results

```r
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.

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)
```

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

```
dim(x)
```

```
[1] 17  5
```

You can use dim(x), or nrow(x) and ncol(x).

I need to fix that first column…

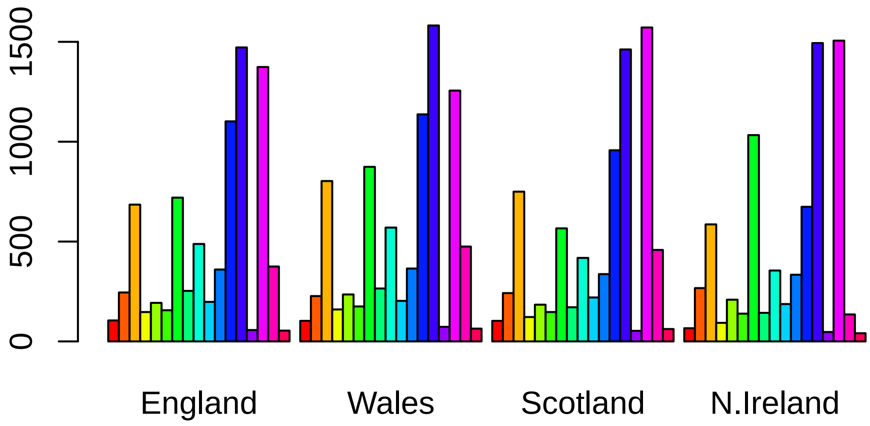
```
rownames(x) <- x[,1]
x <- x[,-1]
head(x)
```

```
               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
```

> 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?

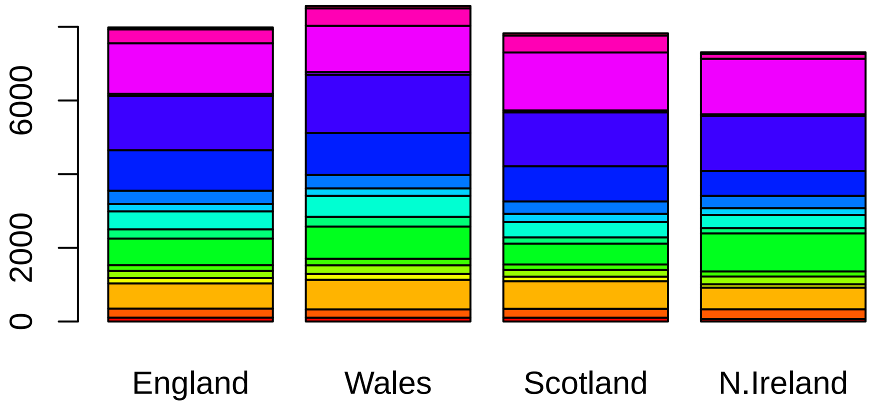
I prefer the row.names=1 argument setting rather than the table manipulation because the table manipulation will occur everytime it is run and rewrite the original table, which can delete columns of data on accident. The row.names=1 argument only occurs once when we load the csv file, so this mistake is prevented.

```
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?

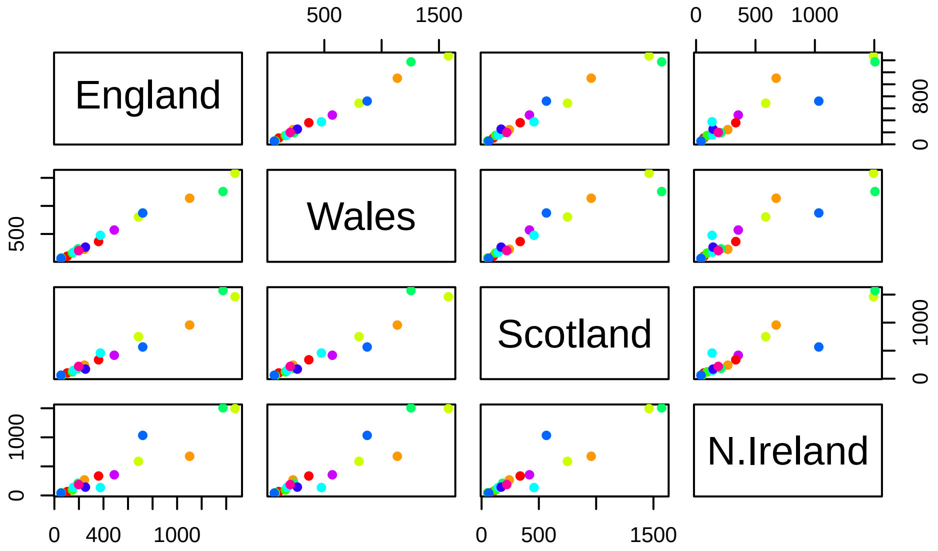
```
barplot(as.matrix(x), beside=FALSE, col=rainbow(nrow(x)), )
```



Changing the `beside` argument will result in it. When false, this arguments makes the bars stack on top of one another.

> 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)
```



The figure shows the plots of every combination of country plotted against one another with respect to their food consumption.

If the point lies on the diagonal for a given plot, it means that similar amounts of that food category was consumed between the two countries. Deviation from the diagonal means that there is more or less of that food category consumed in that country.

> Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set?

It is hard to tell the main differences between N. Ireland and the other countries of the UK in this graphical presentation. We can tell which foods are consumed more and less based on their position relative to the diagonal on the graph.

Principal Component Analysis (PCA)

PCA can help us make sense of these types of datasets. Let's see how it works.

The main function in "base" R is called `prcomp()`. In this case we want to first take the transpose `t()` of our input `x` so the columns are the food types and the countries are the rows.

```
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
```

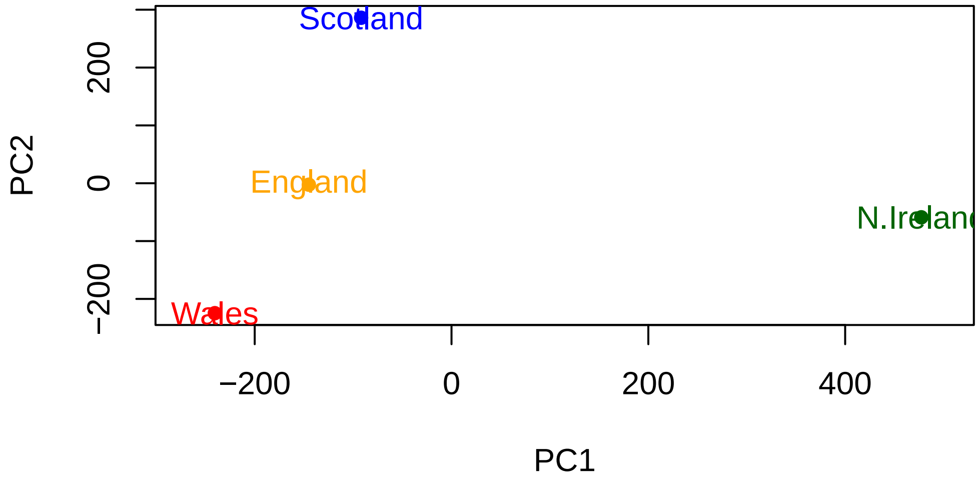
```r
pca <- prcomp(t(x))
summary(pca)
```

```
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
```

> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

> Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```r
plot(pca$x[,1], pca$x[,2],xlab="PC1", col=c("orange","red","blue","darkgreen"),ylab="PC2",
text(pca$x[,1], pca$x[,2], colnames(x), col=c("orange","red","blue","darkgreen"))
```



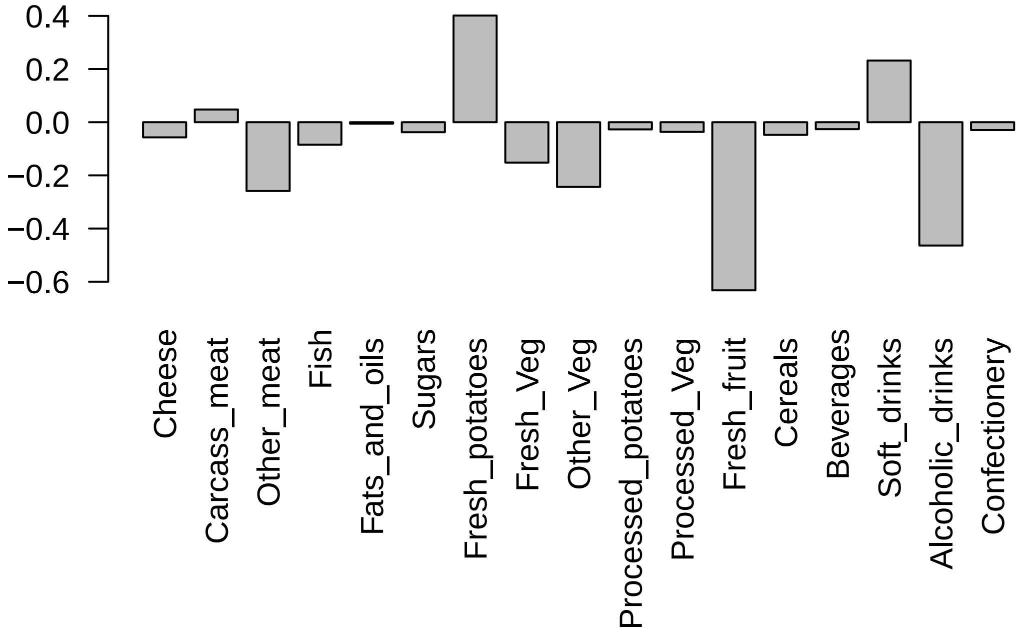
The “loadings” tell us how much the original variables (in our case the foods) contribute to the new variables i.e. the PCs

```
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
```

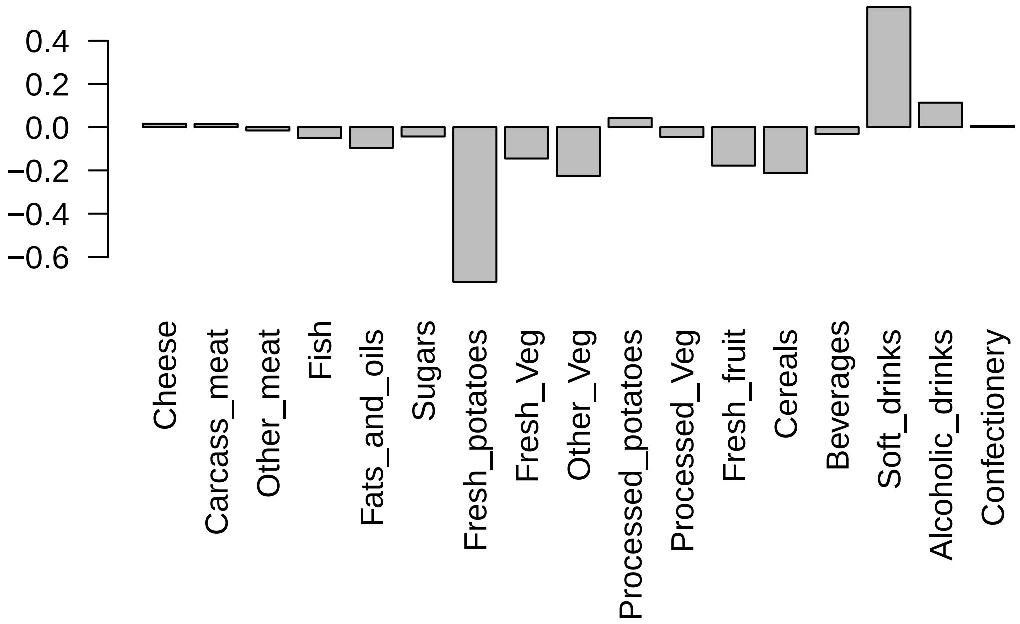
```
Confectionery        -0.029650201  0.005949921 -0.05232164  0.001890737
```

```
## 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 )
```



Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about?

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



The two food groups that feature prominently are Fresh__potatoes and Soft__drinks. PC2 captures the remaining variance that could not be captured from PC1 alone.