Class 7: Machine Learning 1

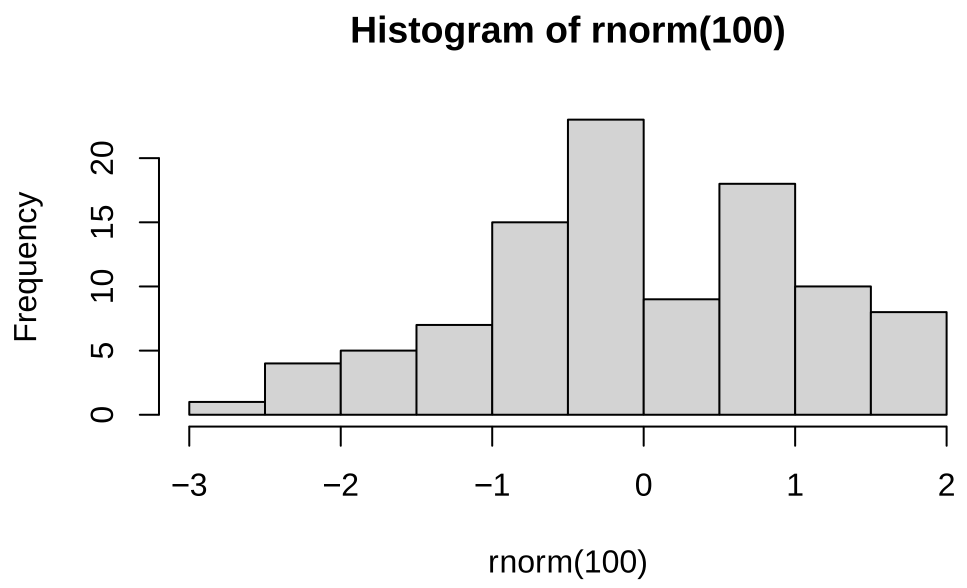
Selma Cifric (PID A69042976)

Machine Learning Introduction

Today, we explore "classical" machine learning approaches. We will start with clustering.

We will make up data to cluster where we known what the answer should be.

```r
hist(rnorm(100))
```

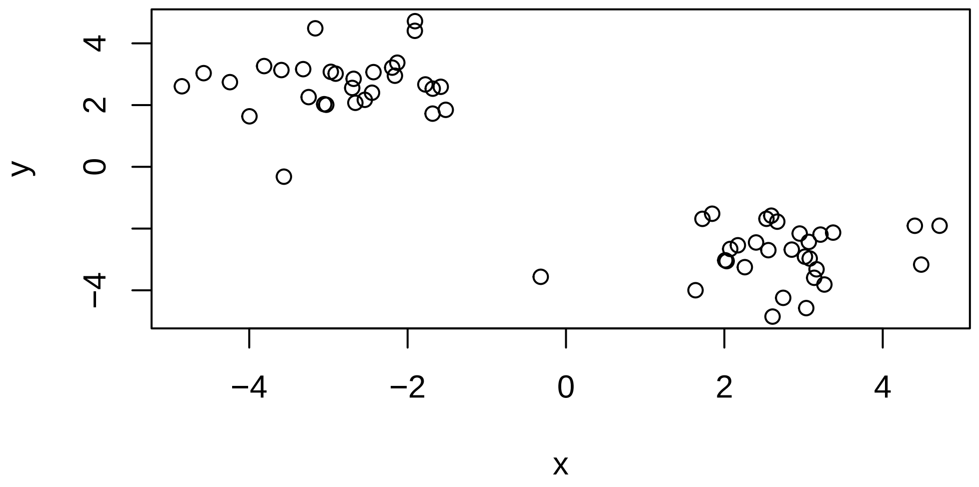


```r
x <- c( rnorm(30, mean = -3), rnorm(30, mean = 3) )
y <- rev(x)

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

```
             x        y
[1,] -2.699522 2.555370
[2,] -2.130546 3.373286
[3,] -1.683024 2.531355
[4,] -1.776043 2.668128
[5,] -3.593881 3.134324
[6,] -3.812172 3.263090
```

We can plot x now with `plot()`:

```
plot(x)
```



The main function in "base" R for K-means clustering is called `kmeans()`

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

> Q. How big are the clusters (i.e. their size)?

```
k$size
```

```
[1] 30 30
```

This tells us that we have two clusters, each containing 30 points.

Q. What clusters do my data points reside in?

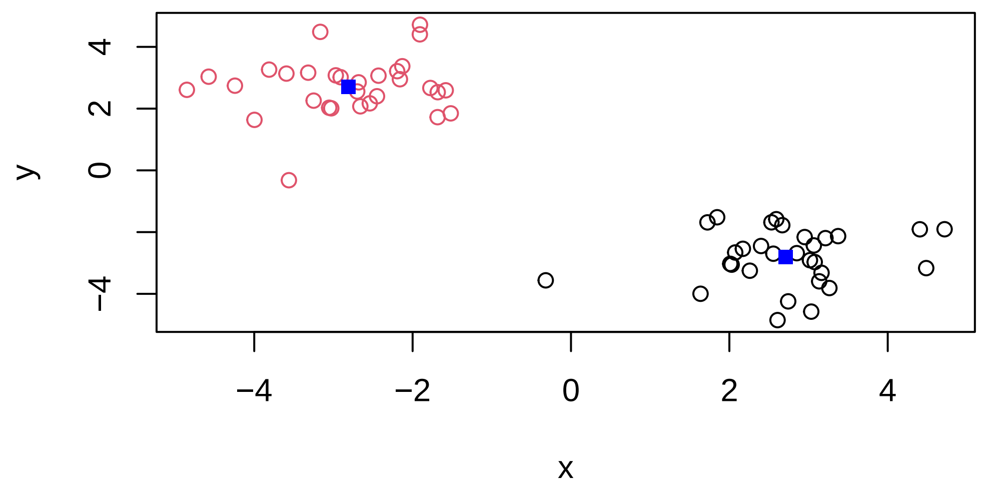
```
k$cluster
```

```
 [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 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
```

Automatically, 1s are assigned to cluster 1 and 2s are assigned to cluster 2.

Q. Make a plot of our data colored by cluster assignment (i.e. Make a result figure).

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

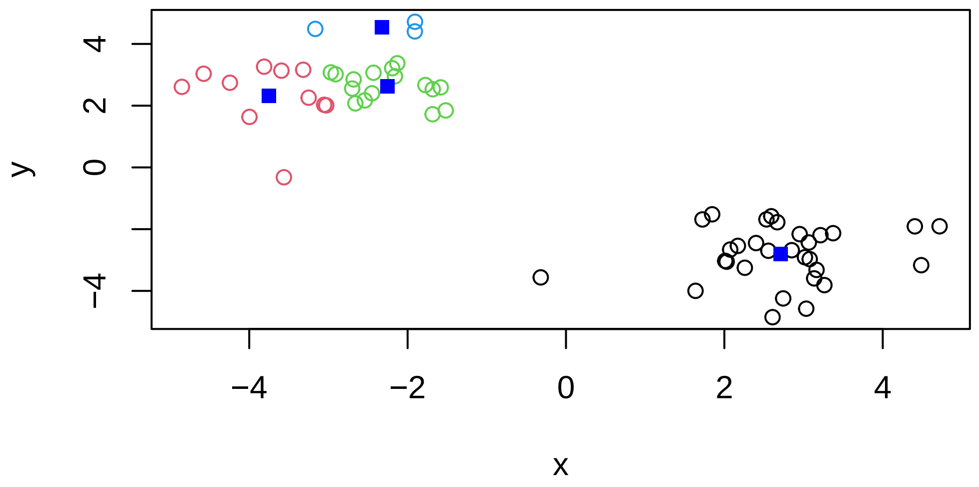


Q. Cluster with k-means into 4 clusters and plot your results.

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

```
K-means clustering with 4 clusters of sizes 30, 11, 16, 3

Cluster means:
          x         y
1  2.709363 -2.811428
2 -3.753393  2.323706
3 -2.254589  2.631940
4 -2.327364  4.536364

Clustering vector:
 [1] 3 3 3 3 2 2 3 2 2 2 3 2 2 3 3 3 2 3 4 4 2 3 3 3 2 3 3 2 3 4 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] 49.789784 14.357890  7.153579  1.108214
 (between_SS / total_SS =  92.9 %)

Available components:

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

```r
plot(x, col = new$cluster)
points(new$centers, col="blue", pch=15)
```



Q. Run kmeans with center (i.e. values of k=1:6)

```
new$tot.withinss
```

```
[1] 72.40947
```

```
new1 <- kmeans(x, centers=1)$tot.withinss
new2 <- kmeans(x, centers=2)$tot.withinss
new3 <- kmeans(x, centers=3)$tot.withinss
new4 <- kmeans(x, centers=4)$tot.withinss
new5 <- kmeans(x, centers=5)$tot.withinss
new6 <- kmeans(x, centers=6)$tot.withinss

ans <- c(new1, new2, new3, new4, new5, new6)
```

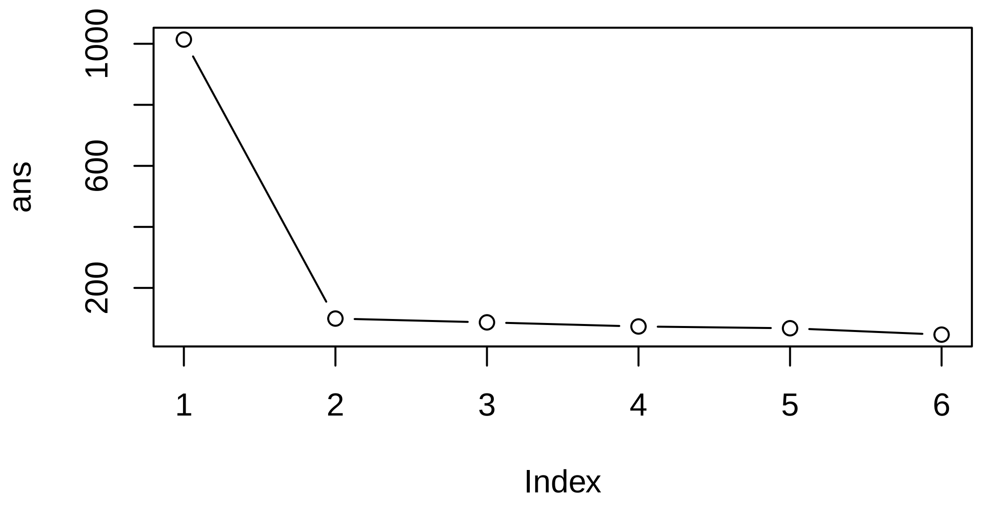
OR use a `for()` loop:

```
ans <- NULL
for (i in 1:6) {
  ans <- c(ans, kmeans(x, centers=i)$tot.withinss)
}
ans
```

```
[1] 1013.95362   99.57957   87.15523   73.65831   67.88330   46.89273
```

Then, we plot it:

```
plot(ans, typ="b")
```



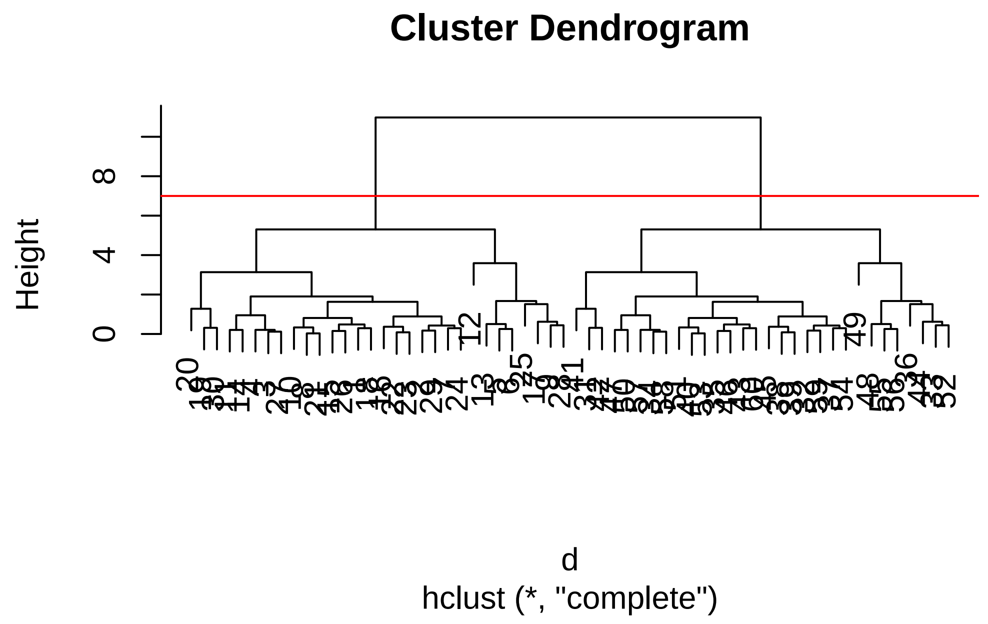
Hierarchial clustering

The main function is the "base" R for this is called `hclust()`

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

```
Call:
hclust(d = d)

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

```r
plot(hc)
abline(h=7, col="red")
```



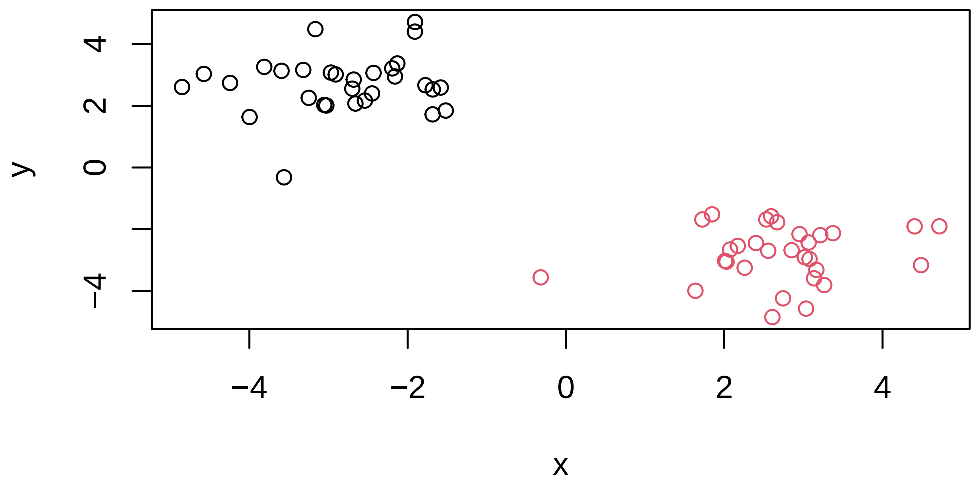
To obtain clusters from out `hclust()` result object **hc** we "cut" the tree to yield fifferent sub branches. For hsi we use the `cuttree()` function:

```r
grps <- cutree(hc, h=7)
grps
```

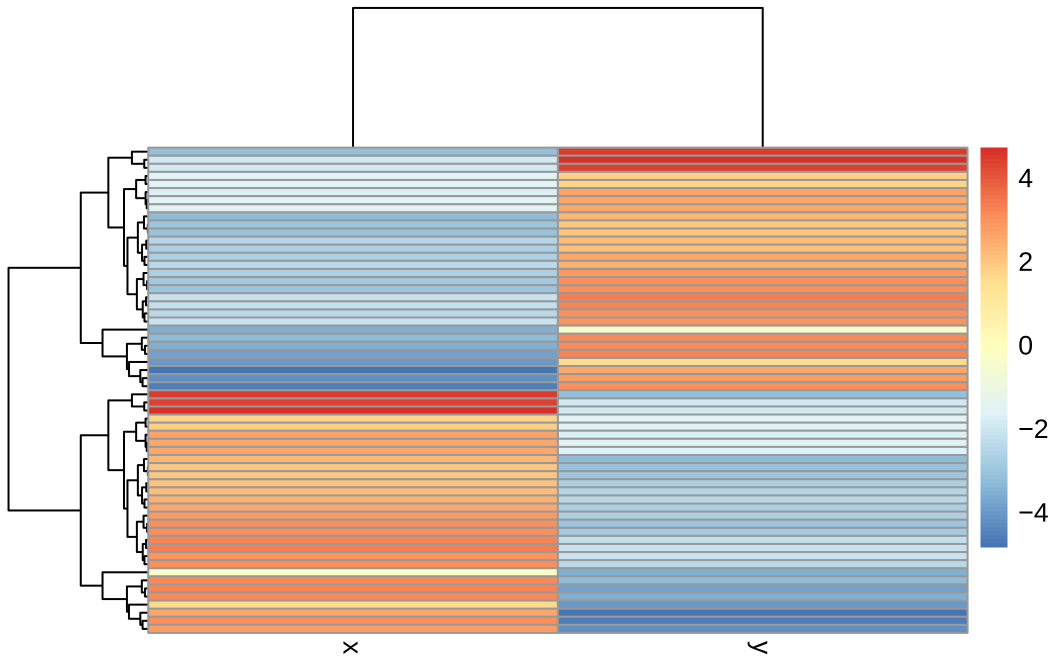
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Results figure:

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



```r
library(pheatmap)
pheatmap(x)
```



Principal Component Analysis (PCA)

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

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

```
[1] 17  5
```

```r
# View(x)
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
```

```r
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
```

```r
dim(x)
```

```
[1] 17  4
```

Alternative approach:

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

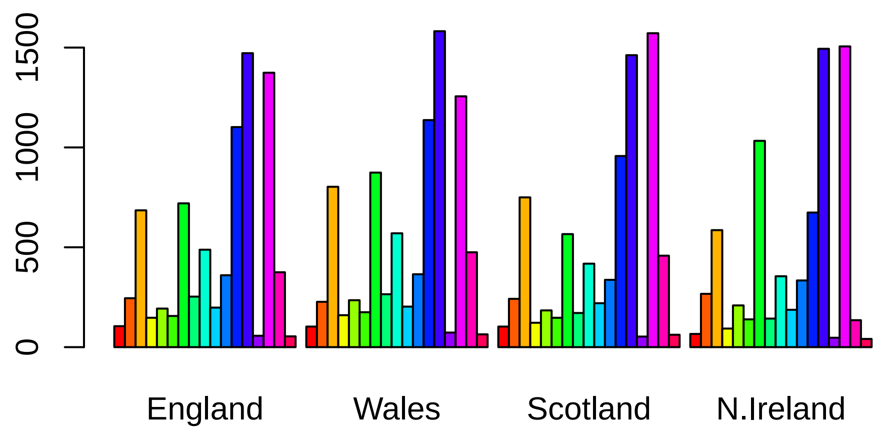
```
               England Wales Scotland N.Ireland
Cheese             105   103      103        66
Carcass_meat       245   227      242       267
Other_meat         685   803      750       586
Fish               147   160      122        93
Fats_and_oils      193   235      184       209
Sugars             156   175      147       139
```

Spotting major differences and trends

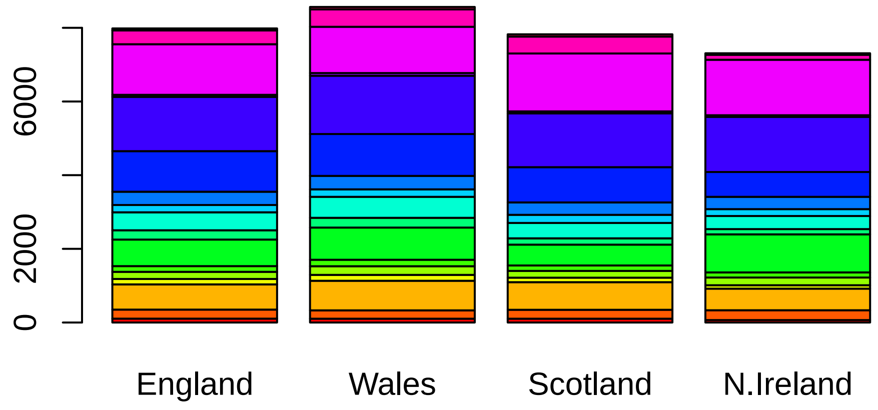
```
rainbow(nrow(x))
```

```
 [1] "#FF0000" "#FF5A00" "#FFB400" "#F0FF00" "#96FF00" "#3CFF00" "#00FF1E"
 [8] "#00FF78" "#00FFD2" "#00D2FF" "#0078FF" "#001EFF" "#3C00FF" "#9600FF"
[15] "#F000FF" "#FF00B4" "#FF005A"
```

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

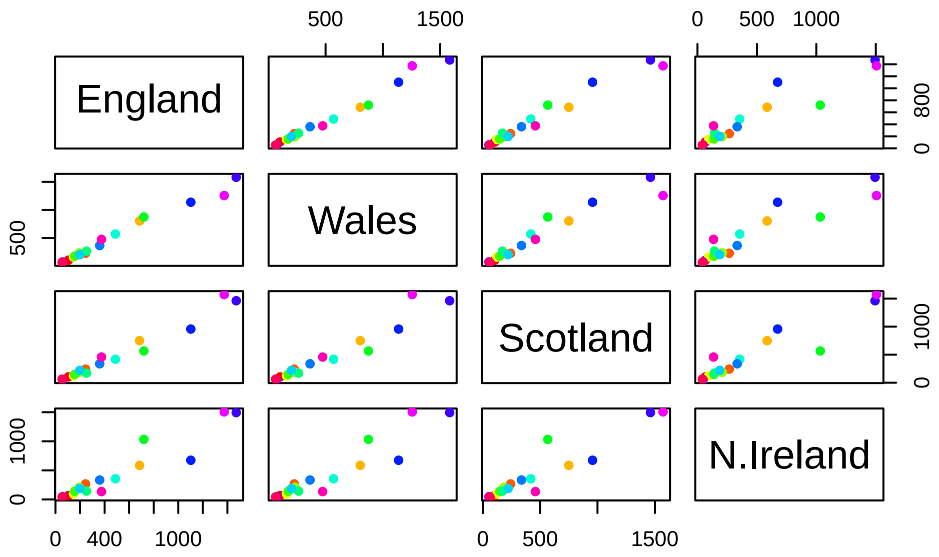


```r
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



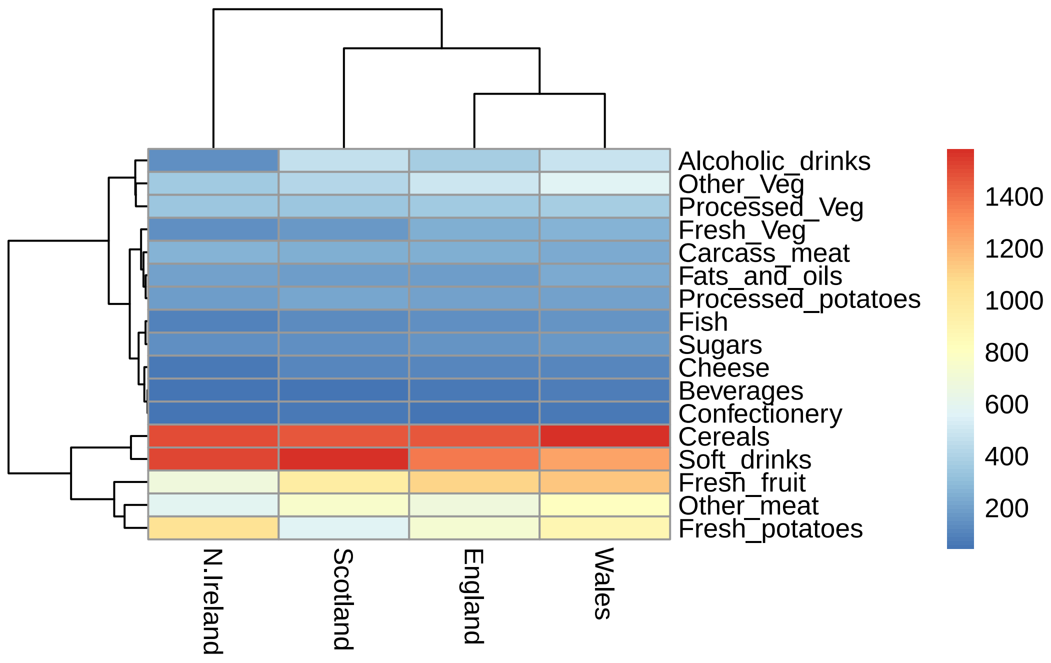
Q5: We can use the `pairs()` function to generate all pairwise plots for our countries. Can you make sense of the following code and resulting figure? What does it mean if a given point lies on the diagonal for a given plot?

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



If a point lies off the diagonal line, it means it's more present in one variable over the other, whichever axis its closer to.

```
pheatmap(as.matrix(x))
```



> Q6. Based on the pairs and heatmap figures, which countries cluster together and what does this suggest about their food consumption patterns? Can you easily tell what the main differences between N. Ireland and the other countries of the UK in terms of this data-set?

While we do see similarities between Wales and England, it is quite difficult to tell what is going on in the dataset.

PCA to the rescue

The main function in "base" R for PCA is called `prcomp()`. We want to look at the foods data so we can transpose our data frame with `t(x)` to get the foods in the columns, then run PCA code.

```
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
```

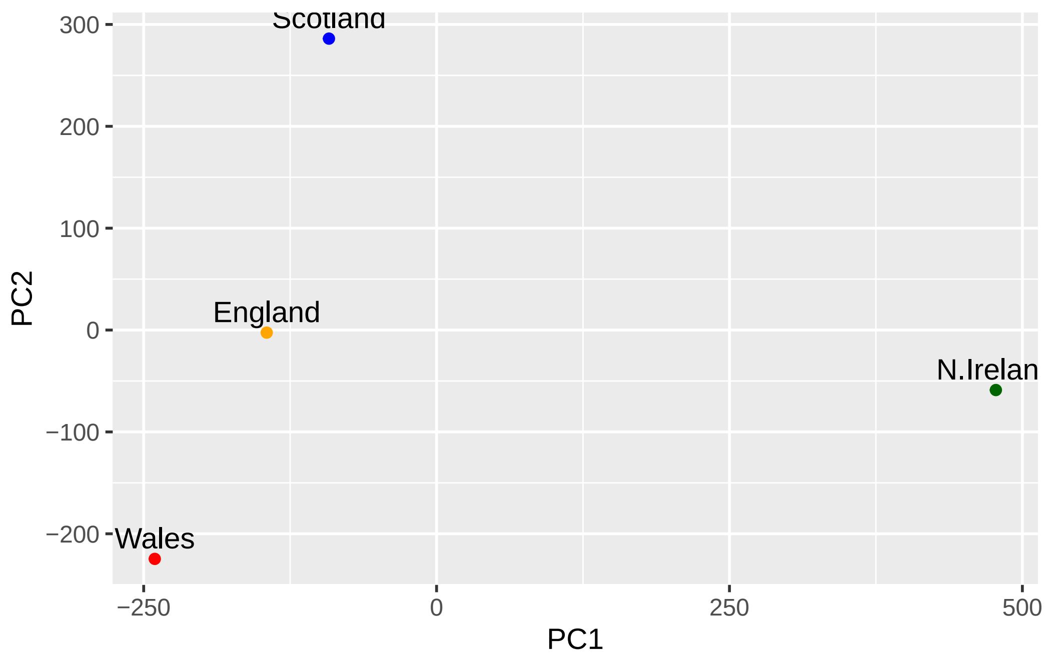
Our result object is called `pca`, which is a list object and can be called with a `$x` component.

```
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
```

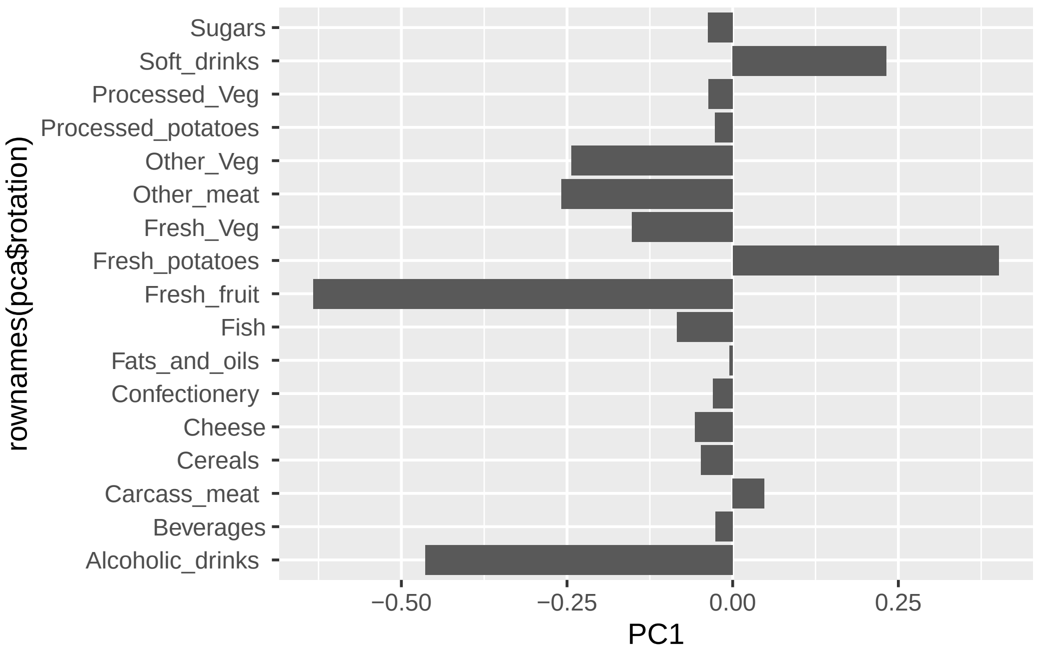
We want to plot PC1 (x-axis) and PC2 (y-axis):

```
library(ggplot2)

cols <- c("orange","red","blue","darkgreen")
ggplot(pca$x) +
  aes(PC1, PC2, label=rownames(pca$x)) +
  geom_point(col=cols) +
  geom_text(vjust = -0.5)
```



Another major result of PCA is the so-called "variable loadings" or `$rotation` that tells us how the original variables (foods) contribute to the new axis (PCs).

```
ggplot(pca$rotation) +
  aes(PC1, rownames(pca$rotation)) +
  geom_col()
```



0 in PCA plots means average, and movement in +direction means that variable is more dominant for that group.