Class 07 : Machine Learning 1

Kyle Wittkop (A18592410)

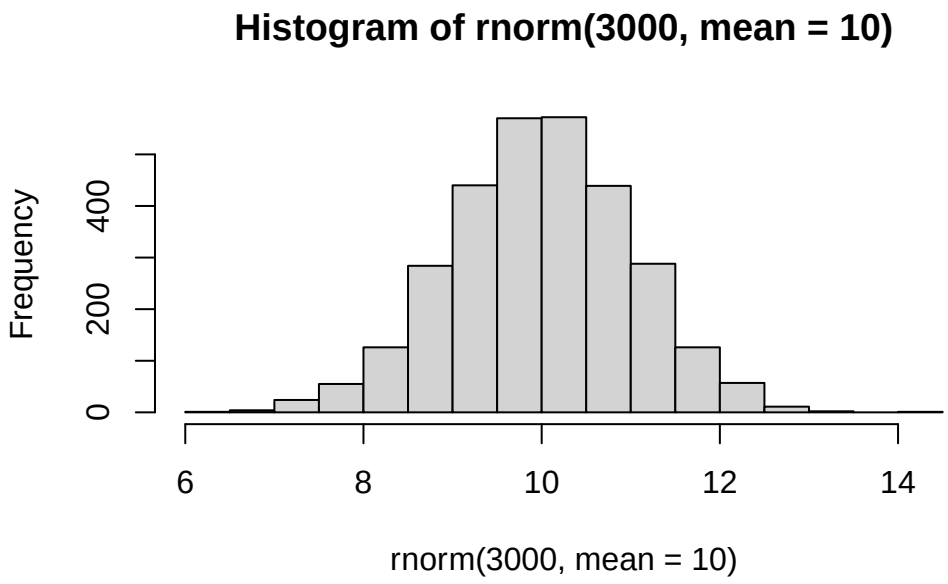
Table of contents

Backround

Today we will begin our exploration of important machine learning methods, as a focus on **clustering** and **dimensionality reduction**.

To start testing these methods, lets make up some sample data dots to cluster where we know what the answer should be.

```
hist(rnorm(3000, mean=10))
```



Q. Can you generate 30 numbers centered around +3 taken atrandom drom a normal distribution

```
rnorm(30, mean=3)
```

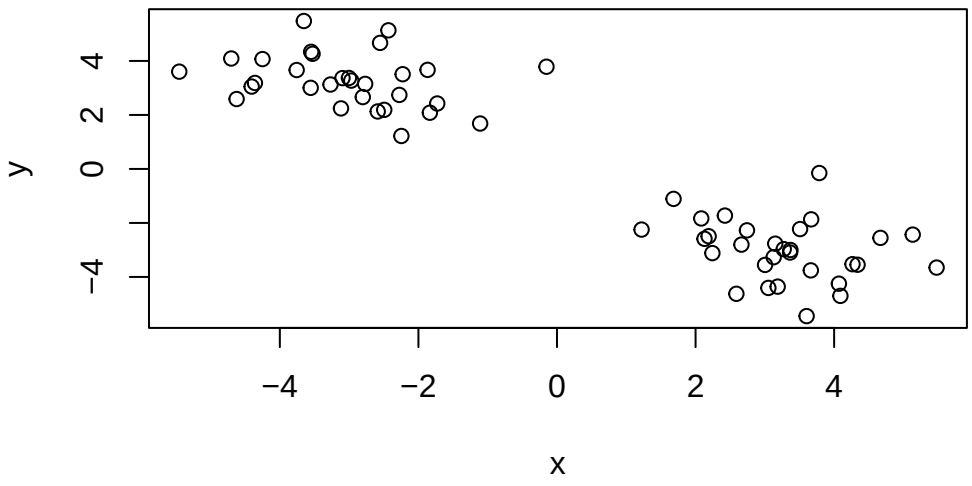
```
 [1] 3.573519 3.128351 3.555634 4.049601 1.830289 1.206520 3.492734 1.933054
 [9] 4.190879 2.823972 1.218854 3.594317 1.426486 1.563269 2.388148 3.786902
[17] 2.569394 3.186117 4.055719 2.732841 1.534143 3.430038 3.064831 1.347411
[25] 3.015511 2.325580 4.872557 3.404735 4.887299 3.779791
```

```
rnorm(30,mean=-3)
```

```
 [1] -3.1029469 -3.6234436 -2.7447262 -2.2855722 -3.5020828 -3.0139025
 [7] -1.4996169 -3.1542440 -4.3546607 -4.6675254 -3.5769224 -3.0029604
[13] -2.6449638 -2.5976411 -3.1652244 -2.6475485 -4.9079405 -3.8105808
[19] -3.8134771 -3.2419254 -1.9414440 -1.0507329 -2.3860012 -3.7321387
[25] -3.0312606 -3.1778409 -2.4186611 -1.1865320 -1.8811112 -0.9851372
```

```
tmp <- c(rnorm(30, mean=3),
rnorm(30,mean=-3))
```

```r
x <-cbind(x=tmp, y=rev(tmp))

plot(x)
```



K-means, cluster

The main functon in Base R, for K means clustering is called `K means()`. Lets try it out.

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

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

Cluster means:
          x         y
1  3.258531 -3.012321
2 -3.012321  3.258531

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 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
```

```
Within cluster sum of squares by cluster:
[1] 65.75109 65.75109
 (between_SS / total_SS =  90.0 %)

Available components:

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

Q. What component of your result object has the cluster centers

```
k$centers
```

```
          x         y
1  3.258531 -3.012321
2 -3.012321  3.258531
```

Q. What component of your result object has the cluster size

```
k$size
```

```
[1] 30 30
```

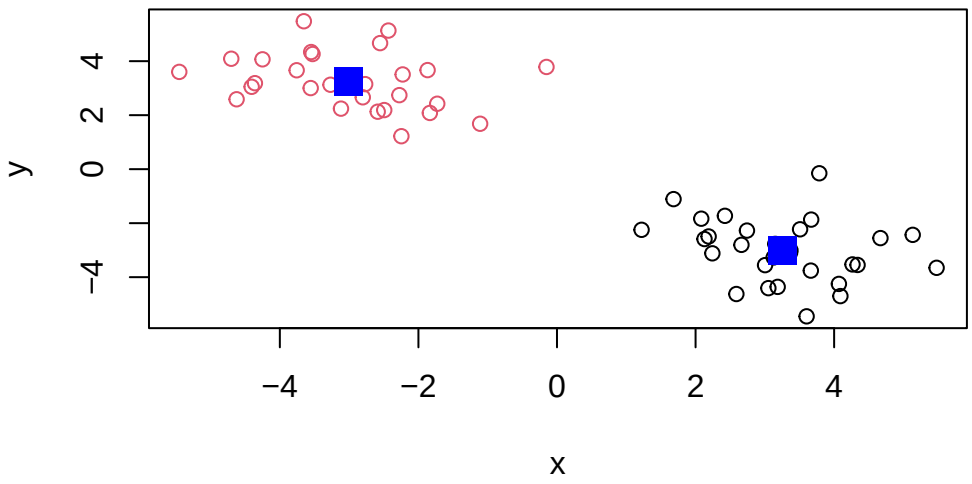
Q. What component of your result object has the cluster membership vector (i.e the main clustering result: which poins are in which cluster)

```
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 2
```

Q. Plot the result of cluster (i.e)

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



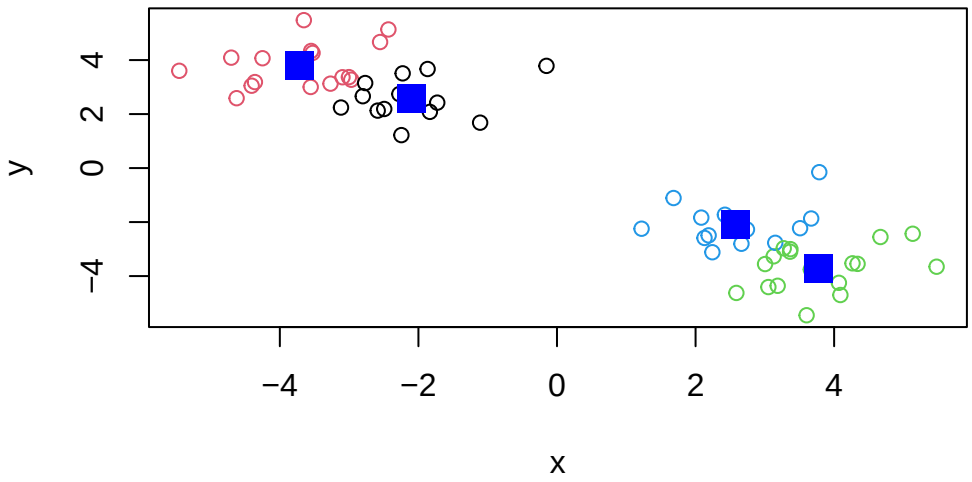
Q. Can you run k means again and cluster into 4 groups and plot the results

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

```
K-means clustering with 4 clusters of sizes 13, 17, 17, 13

Cluster means:
          x         y
1 -2.093082  2.575785
2 -3.715269  3.780631
3  3.780631 -3.715269
4  2.575785 -2.093082

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

Within cluster sum of squares by cluster:
[1] 14.73038 20.94155 20.94155 14.73038
 (between_SS / total_SS =  94.6 %)

Available components:
```

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

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



Key point K-means will always return clustering that we ask for, this is the k or `centers` in K means.

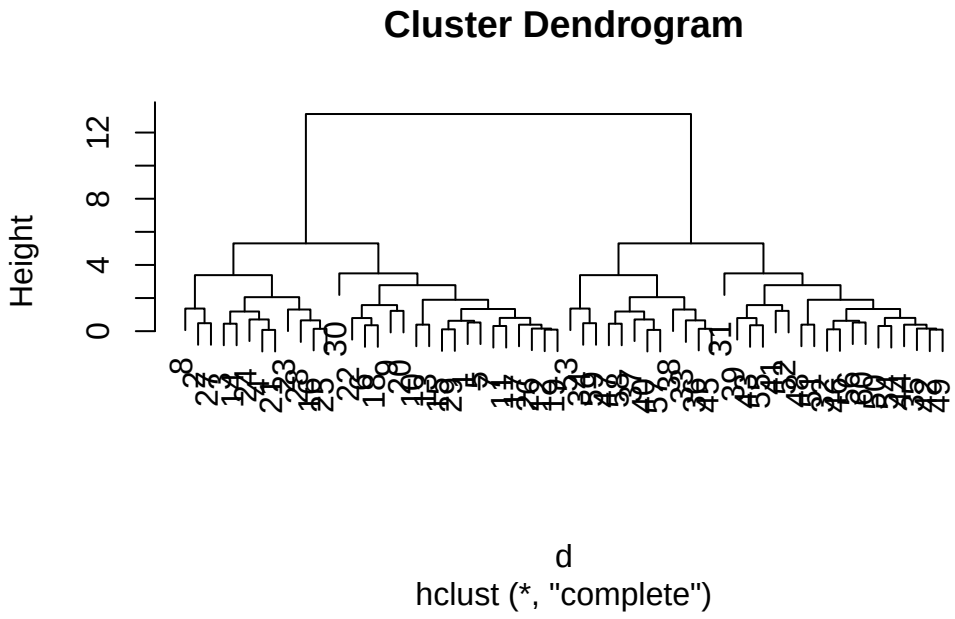
```
k$tot.withinss
```

```
[1] 71.34387
```

Hierarchical clustering

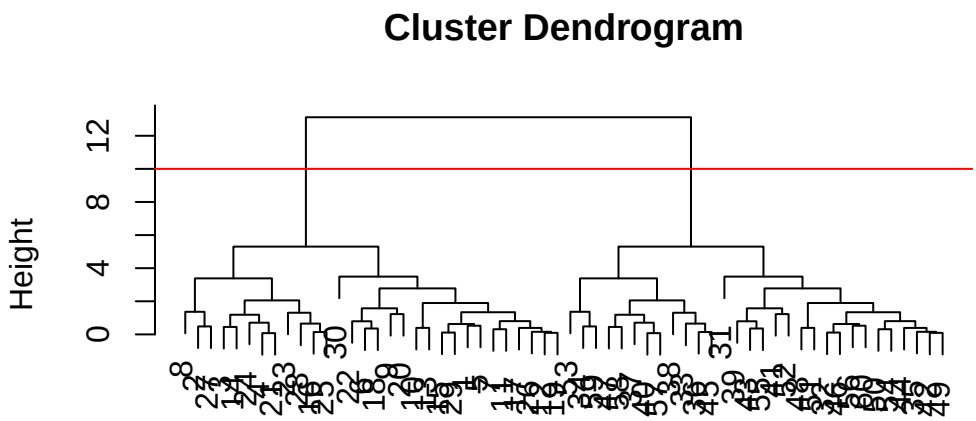
The main function to do this in base R is called hclust `hclust()`. One of the main differences with respect to the k means function, is that you cannot put your input data directly. it needs a distance matrix or a dissimilarity matrix. we can get this from lots of places including the dist function,

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



We can cut the deprogram, at a given height to yield our clusters

```
plot(hc)
abline(h=10,col="red")
```



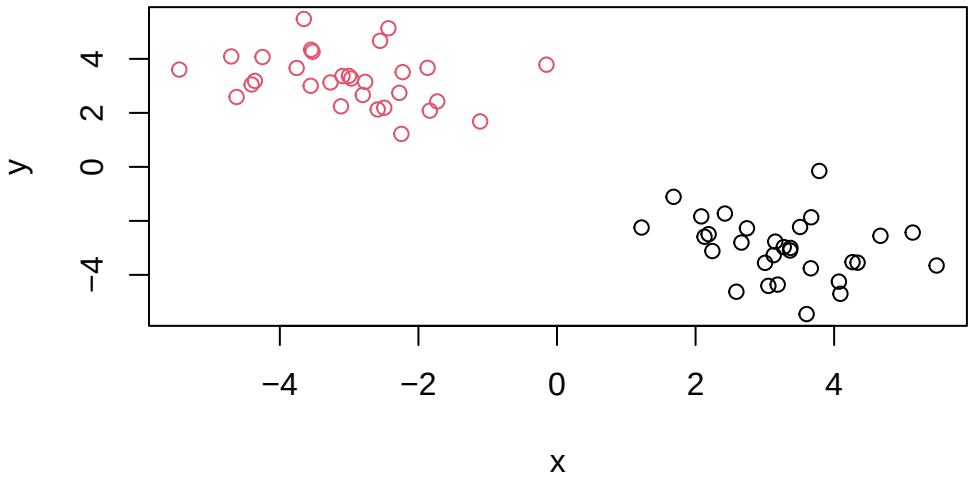

```
grps<-cutree(hc, h=10)
```

```
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 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
```

Q, Plot out data 'x' clored by clustering resulting from hclust (), and cutree

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



Principal component analysis (PCA)

PCA is a popular nationality reduction technique thats widly used in bioinformatics

PCA of UK food data

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

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

```
[1] 17  5
```

It looks like the row names were not set properly. We can fix this.

```
rownames(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
```

A Better way to do this is to fix the row names assignment at import times :

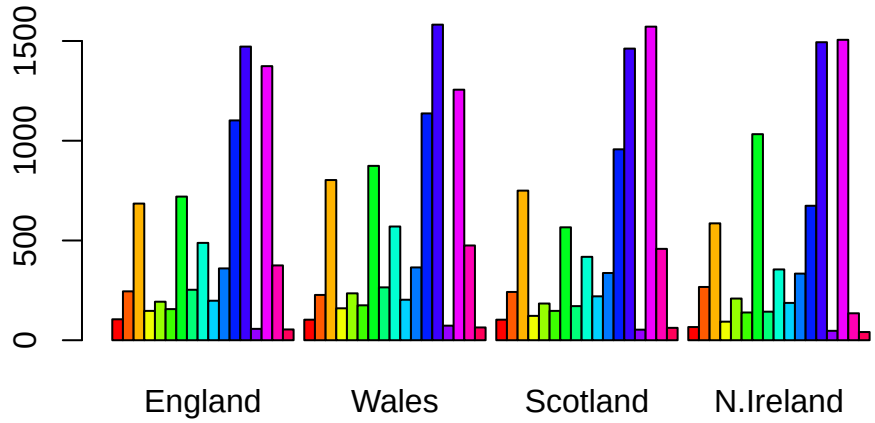
```
x <-read.csv(url, row.names = 1)
x
```

```
                  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
```

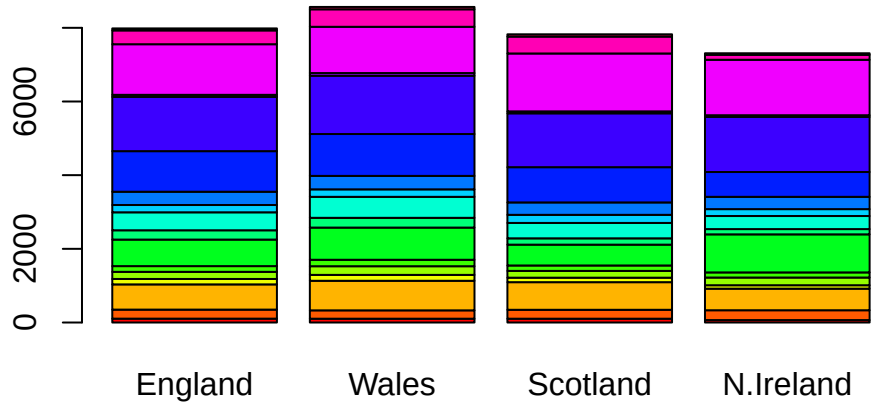
Cereals	1472	1582	1462	1494
Beverages	57	73	53	47
Soft_drinks	1374	1256	1572	1506
Alcoholic_drinks	375	475	458	135
Confectionery	54	64	62	41

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

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

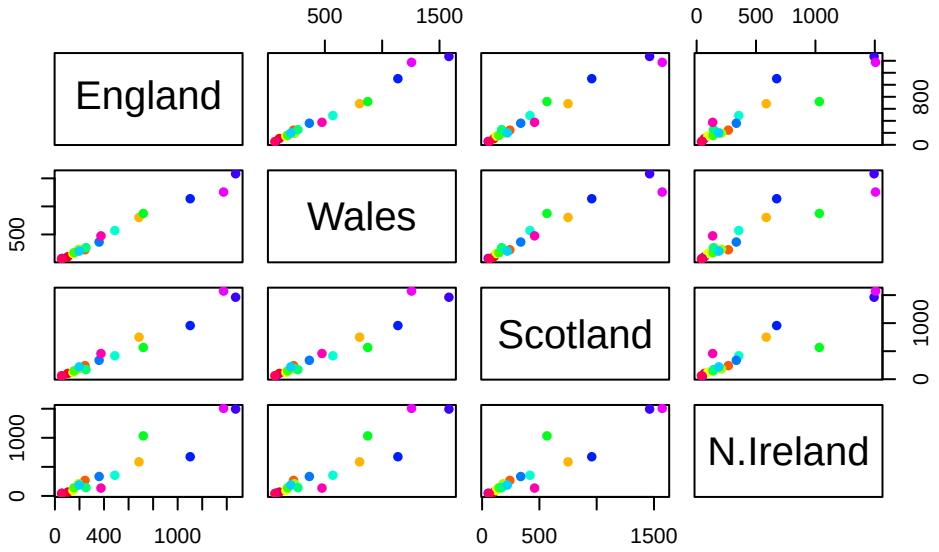


```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



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?

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

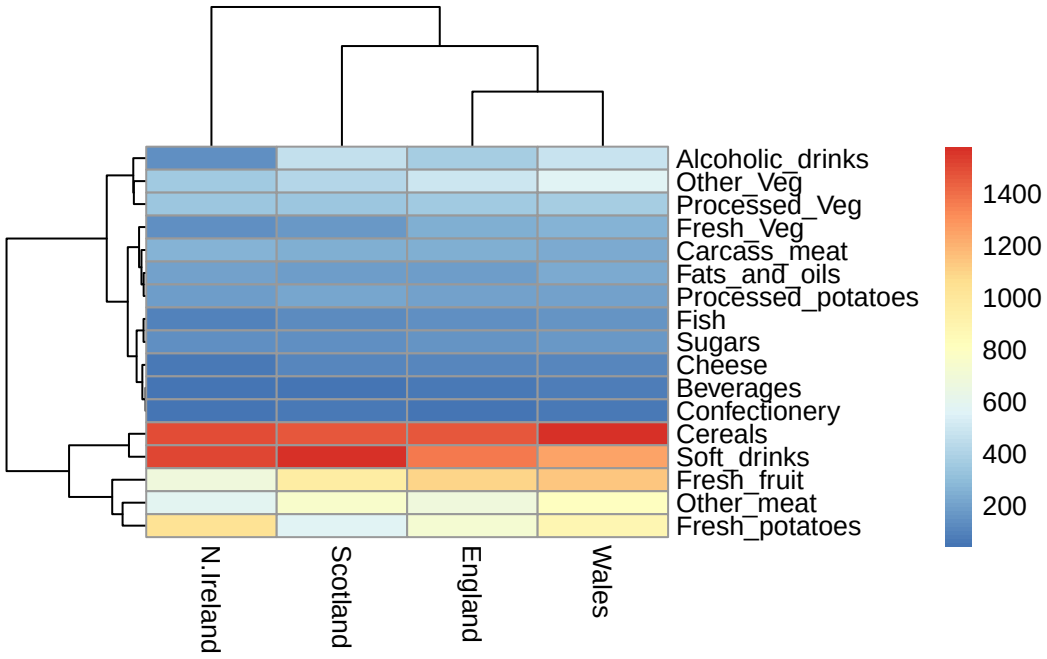


each point represents the data surrounding a food consumed by the 4 countries. If a given point lies on the diagonal for a given plot, it showed that the point is shared in value between the two countries plotted against each other. If it is off the diagonal, it means there is different values for these foods in these two countries.

Heatmap

we can install the ***pheatmap*** package with the install.packages command that we used the previously. Remember that we always run this in console and not a code chunk in our quarto document

```
library(pheatmap)

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?

Based on the pairs and heat map figures, it appear that England, Wales, and Scotland cluster together suggesting their food consumption patterns are similar. We can fairly easily tell the different between Ireland and the other countries, however the pairs shows this difference more effectively as seen in the disorder of points along the diagonal when comparing to the other countries.

Of all these plots, really only the pairs plot was useful. This however took a bit of work to interpretative and will not scale when looking at much bigger data sets.

PCA to the rescue

The main function in "base R" for PCA is called `prcomp()`

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

```
Importance of components:
                            PC1       PC2       PC3      PC4
```

```
Standard deviation     324.1502 212.7478 73.87622 2.7e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.0e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.0e+00
```

> Q. How much variance is captured in the first PC?

67.44% of variance is captured in the first PC

> Q. How many PCs do I need to capture to acheive at least 90% of the total cariance in the data set?

We need to capture 2 PCs in order to achieve a 96.5% of variance shown in cumulative proportions.

> Q Plot out main PCA results Folks can call this different things depending on their field of study for example PC plot.

```
attributes(pca)
```

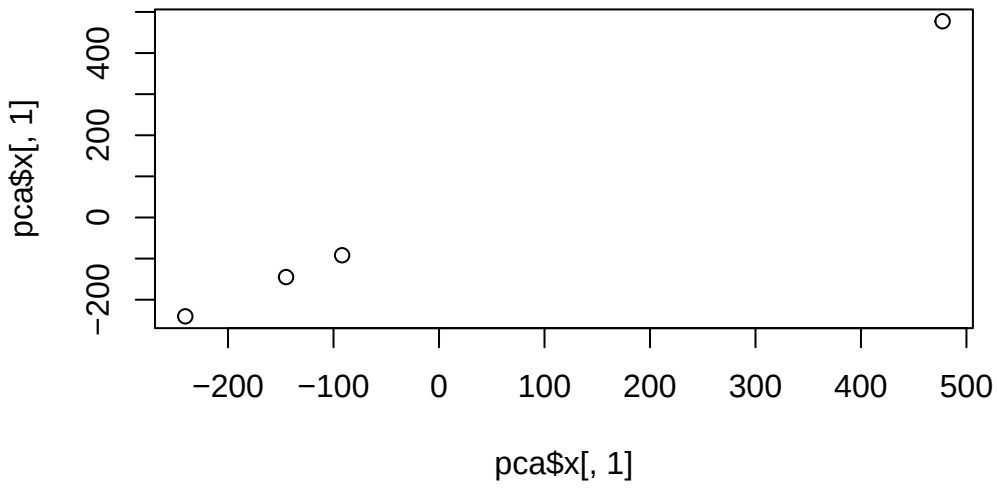
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

To generate our plot we want psa$x

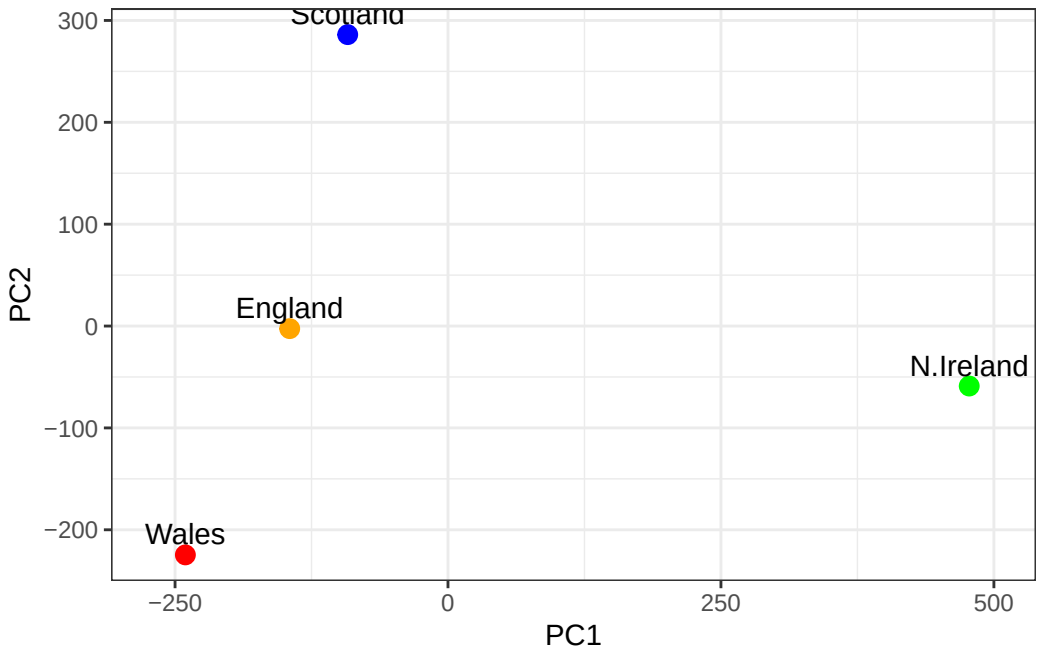
```
pca$x
```

```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945  1.612425e-14
Wales     -240.52915 -224.646925 -56.475555  4.751043e-13
Scotland   -91.86934  286.081786 -44.415495 -6.044349e-13
N.Ireland  477.39164  -58.901862  -4.877895  1.145386e-13
```

```
my_cols <- c("orange","red","blue","green")
plot(pca$x [,1], pca$x [,1])
```



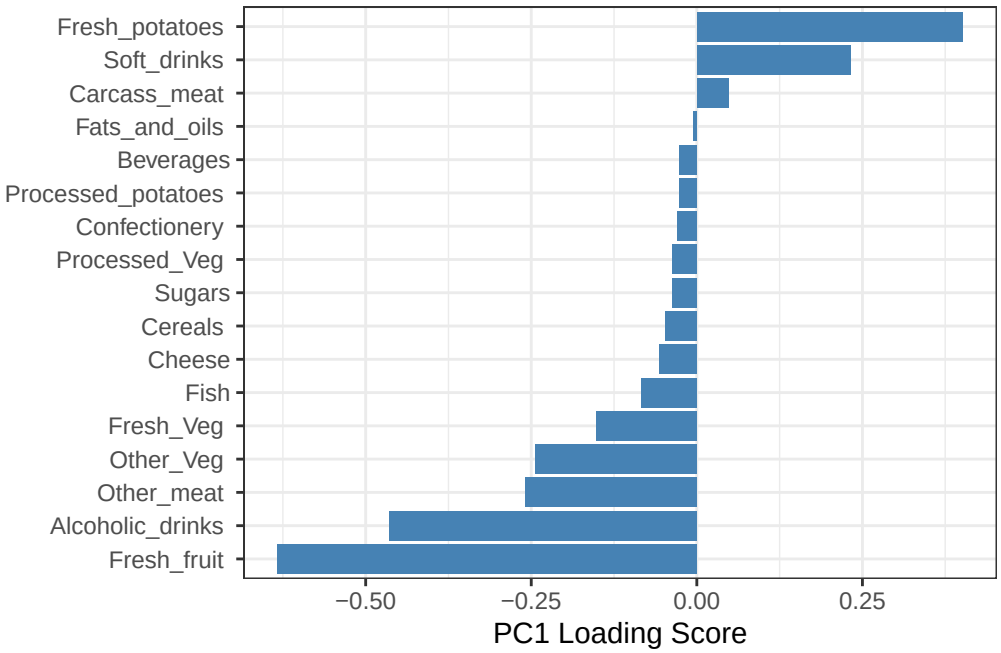

```
library(ggplot2)
ggplot(pca$x) +
  aes(x = PC1, y = PC2, label = rownames(pca$x)) +
  geom_point(size = 3, col=my_cols) +
  geom_text(vjust = -0.5) +
  xlim(-270, 500) +
  xlab("PC1") +
  ylab("PC2") +
  theme_bw()
```



Digging Deeper - Variable loadings

How do the original variables (17 different foods) contribute to our new PCS?

```
ggplot(pca$rotation) +
  aes(x = PC1,
      y = reorder(rownames(pca$rotation), PC1)) +
  geom_col(fill = "steelblue") +
  xlab("PC1 Loading Score") +
  ylab("") +
  theme_bw() +
  theme(axis.text.y = element_text(size = 9))
```


Fresh_potatoes
Soft_drinks
Carcass_meat
Fats_and_oils
Beverages
Processed_potatoes
Confectionery
Processed_Veg
Sugars
Cereals
Cheese
Fish
Fresh_Veg
Other_Veg
Other_meat
Alcoholic_drinks
Fresh_fruit
−0.50
−0.25
0.00
0.25
PC1 Loading Score