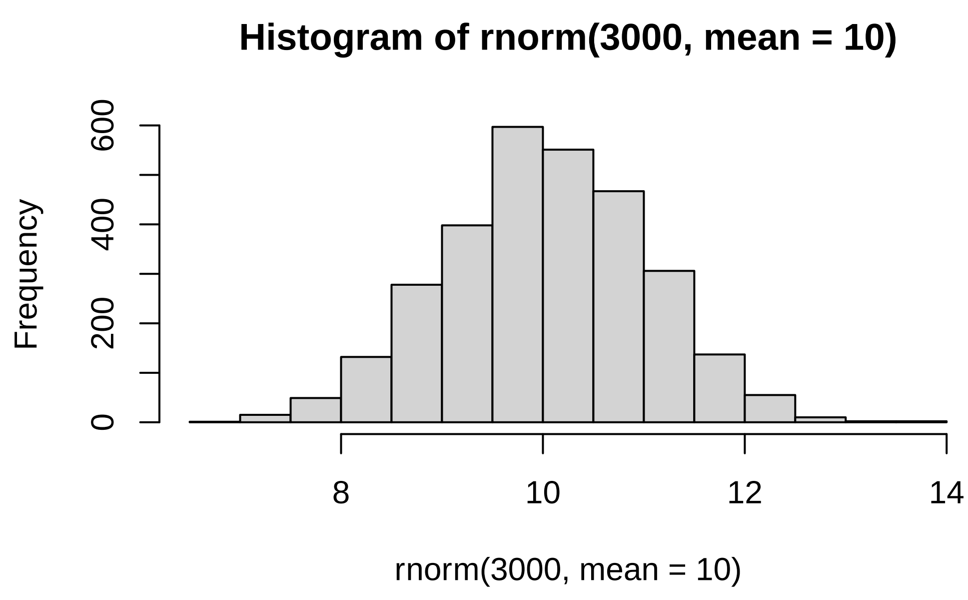
class 07 machine Learining

Bacground

Today we will begin our exploration of important machine learning methods with a focus on **clustering** and **dimensinallity reducation**

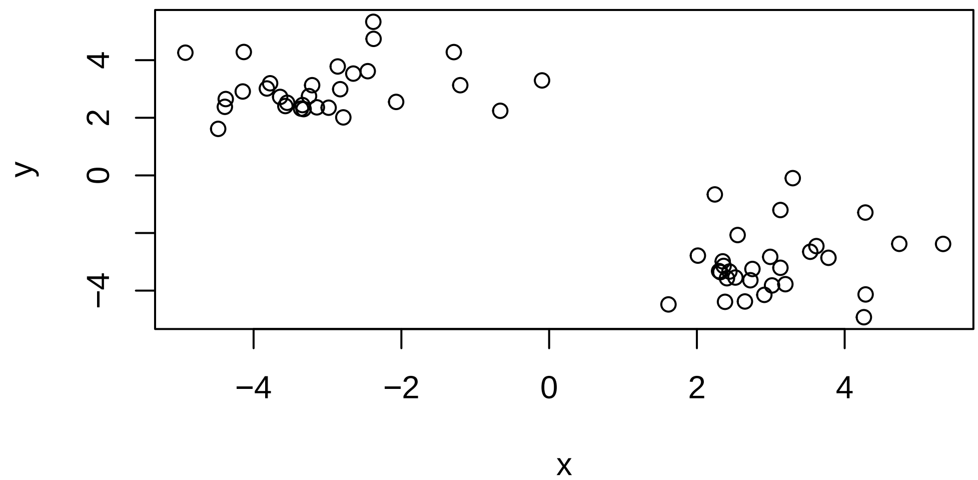
To start testing these methods let's, make up some sample data to cluster where we knoe what the answer should be.

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



Q. Can you generate 30 numbers centered at +3 and 30 numbers at -3 taken random from a normal distribution?

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

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



K-means clustering

The main function "base R" for K-means clustering is called `kmeans()`, lets try it out:

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

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

Cluster means:
          x         y
1 -3.035556  3.037674
2  3.037674 -3.035556

Clustering vector:
```

```
 [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

Within cluster sum of squares by cluster:
[1] 58.99868 58.99868
 (between_SS / total_SS =  90.4 %)

Available components:

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

> Q. what compnet of your kmeans result object has the cluster centers?

```r
k$centers
```

```
          x         y
1 -3.035556  3.037674
2  3.037674 -3.035556
```

> Q. what compnet of your kmeans result object has the cluster size (i.e. how may points are in each cluster?)

```r
k$size
```

```
[1] 30 30
```

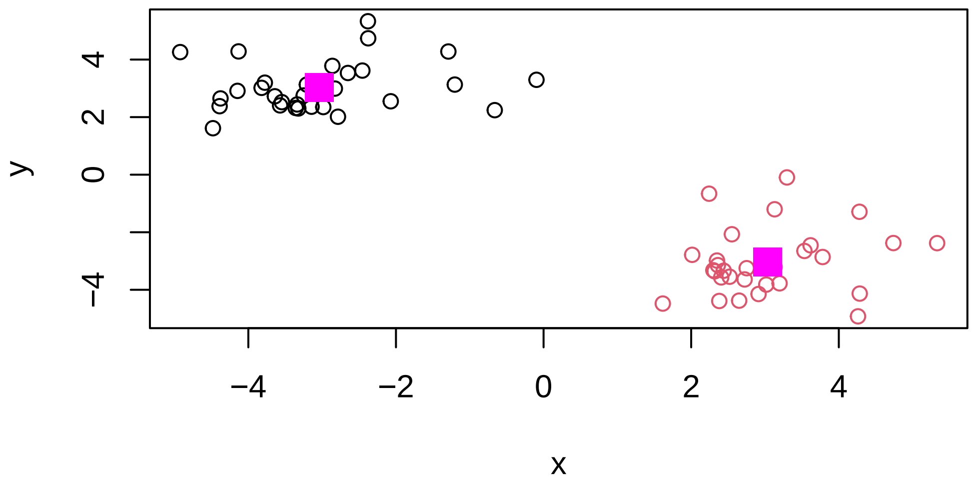
> Q. what compnet of your kmeans result object has the cluster membership vector (i.e. the main clustering result: whcih points are in which cluster?)

```r
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
```

> Q. Plot the results of clustering (i.e. or data colored by the clusterting result) along with the cluster centers

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



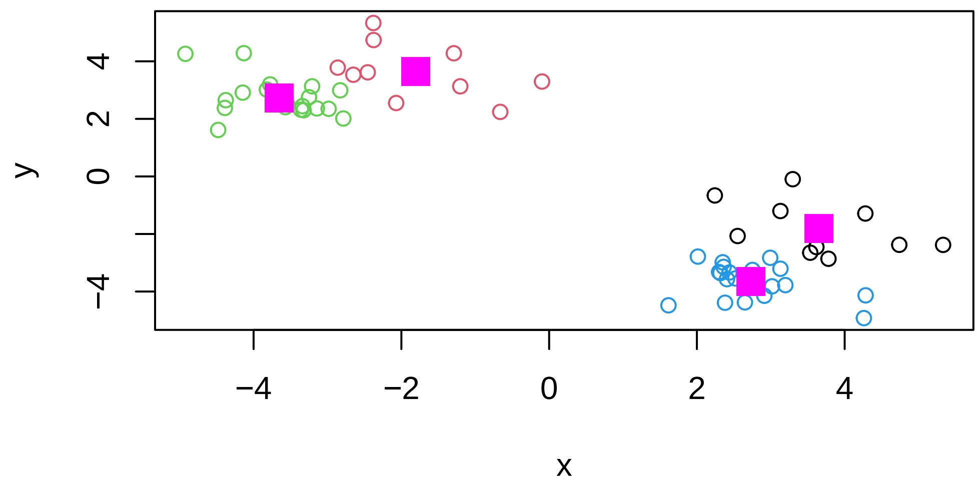
Q. Can you run `kmeans()` and cluster `x` into 4 clusters and plot the results just like we did above with coloring by cluster and the cluster centers shown in blue?

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

```
K-means clustering with 4 clusters of sizes 10, 10, 20, 20

Cluster means:
          x         y
1  3.650537 -1.804441
2 -1.804441  3.650537
3 -3.651113  2.731242
4  2.731242 -3.651113

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

Within cluster sum of squares by cluster:
[1] 15.86523 15.86523 14.76479 14.76479
 (between_SS / total_SS =  95.0 %)
```

```
Available components:

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

```
plot(x, col=k4$cluster)
points(k4$centers, col="magenta", pch=15, cex=2)
```



> **Key point:** Kmeans will always return the clustering that we ask for (this the "K" or "centers" in k-means)!

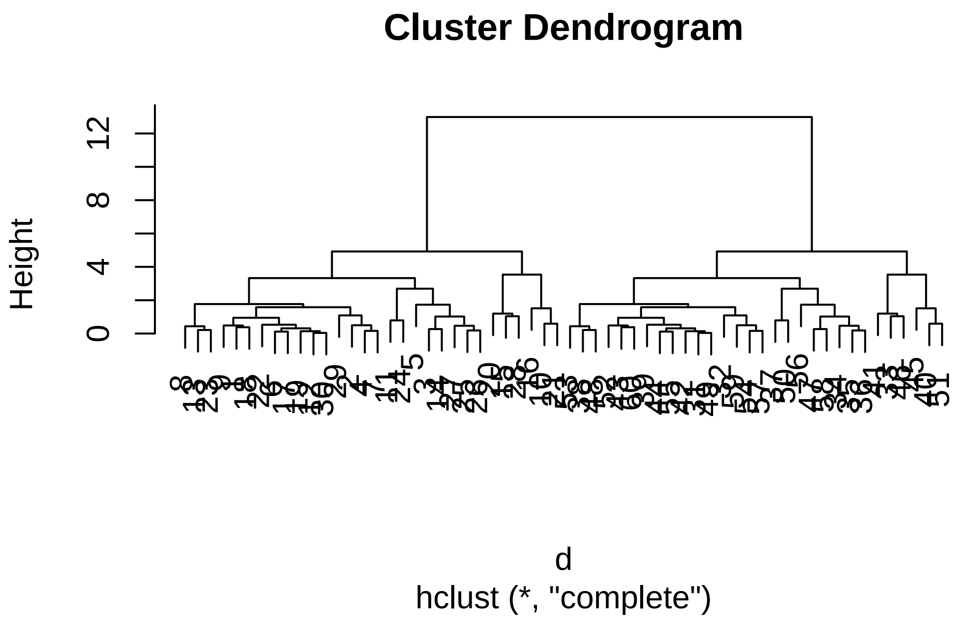
```
k$tot.withinss
```

```
[1] 117.9974
```

Hierarchial clustering

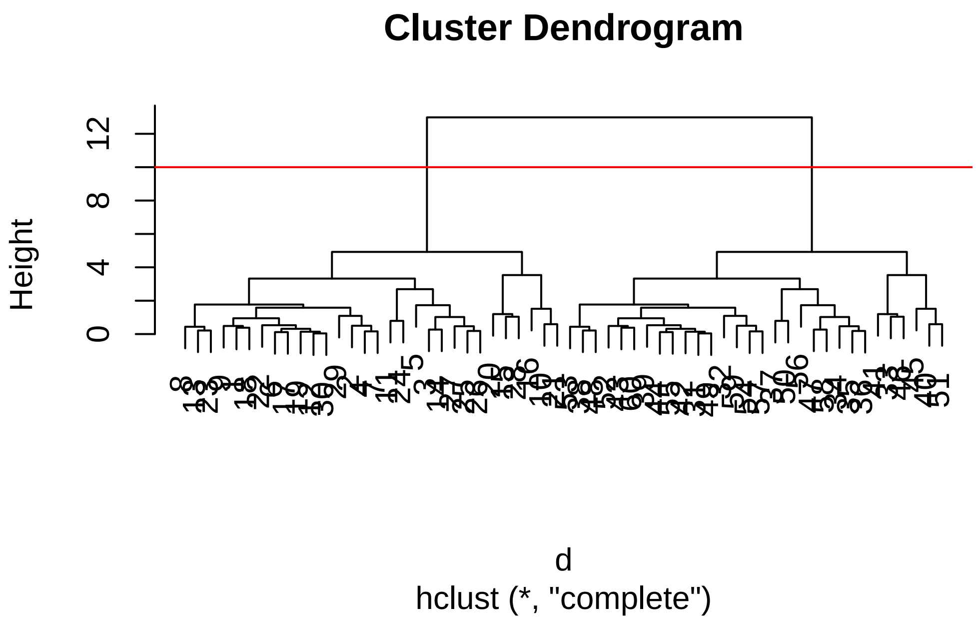
The main function for hierarchical clustering in base R is called `hclust()`. One of the main differences with respect to the `kmeans()` function is that you can not just pass your input data directly to `hclust()` - it needs a "distance matrix" as input. We can get this from losts of places including the `dist()` function.

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



We can cut dendrogram, or tree at a given height to yield our "clusters". For this we use the function `cutree()`

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




```r
cutree(hc, h=10)
```

```
 [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
```

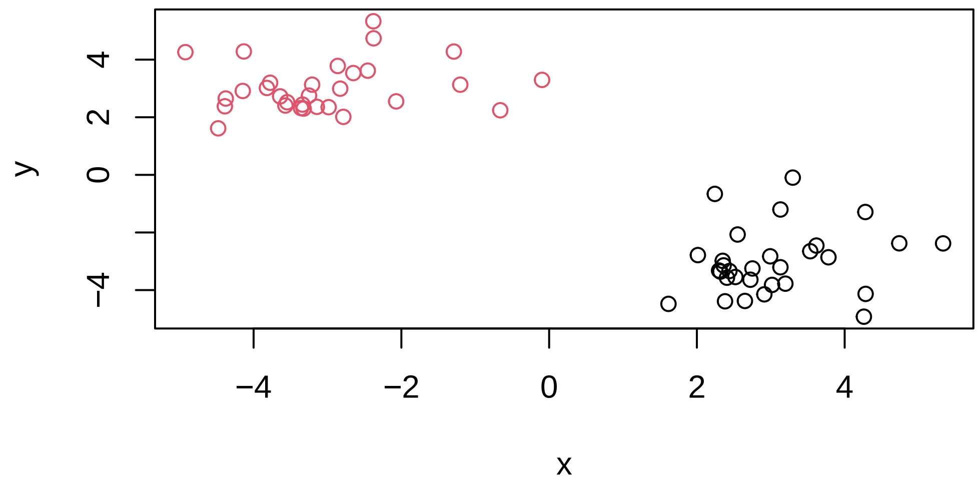
```r
grps <-cutree(hc, h=10)
```

```r
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
```

Q. plot our data `x` colored by the clustering results from `hclust()` and `ctreee()`?

```r
grps <- cutree(hc, h=10)
plot(x, col=grps)
```



Principal Componet Analysis (PCA)

PCA is a popular dimensionality reduction technique that is widely used in bioinformatics.

PCA of UK food data

start the lab sheet…

```
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
```

It looks like the row names are not set prperly. we can fix this.

```r
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 time:

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

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

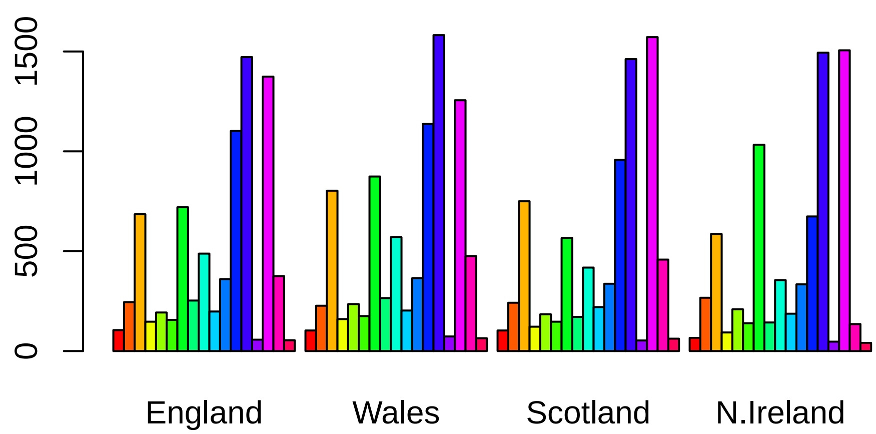
```r
dim(x)
```

```
[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?

- I like the second one the best because it is easy to use and it is not destructive

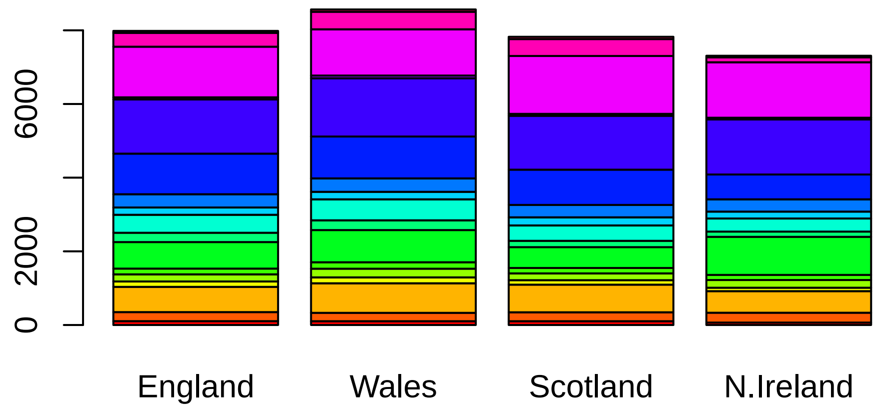
```r
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

- You would chaneg it form T to F

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




```r
dim(x)
```

```
[1] 17  4
```

```r
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
```

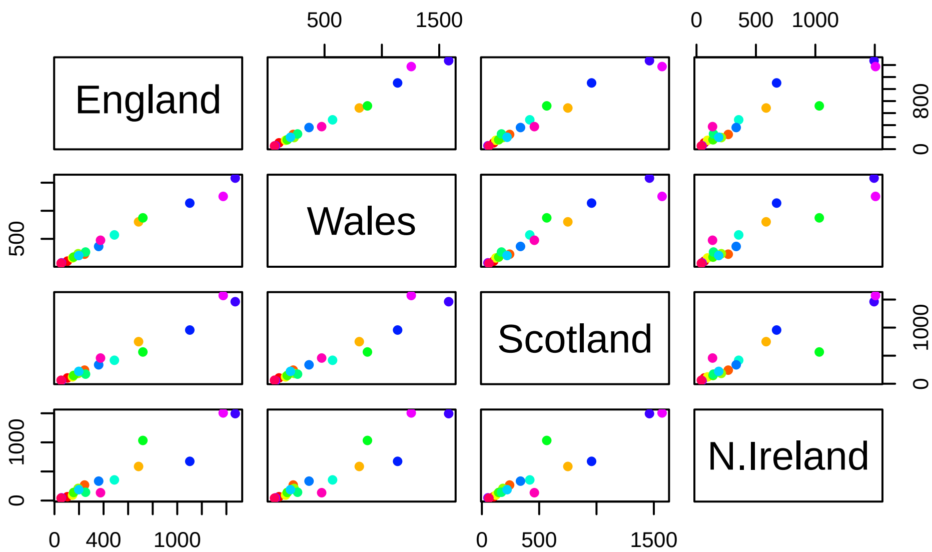
> Q4 Is missing

Pairs plots and heatmaps

> 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?

- if it is high the diagonal it is going to be England because it is more up and England is showing to be the most up. it also shows that England and Wales are very similar.

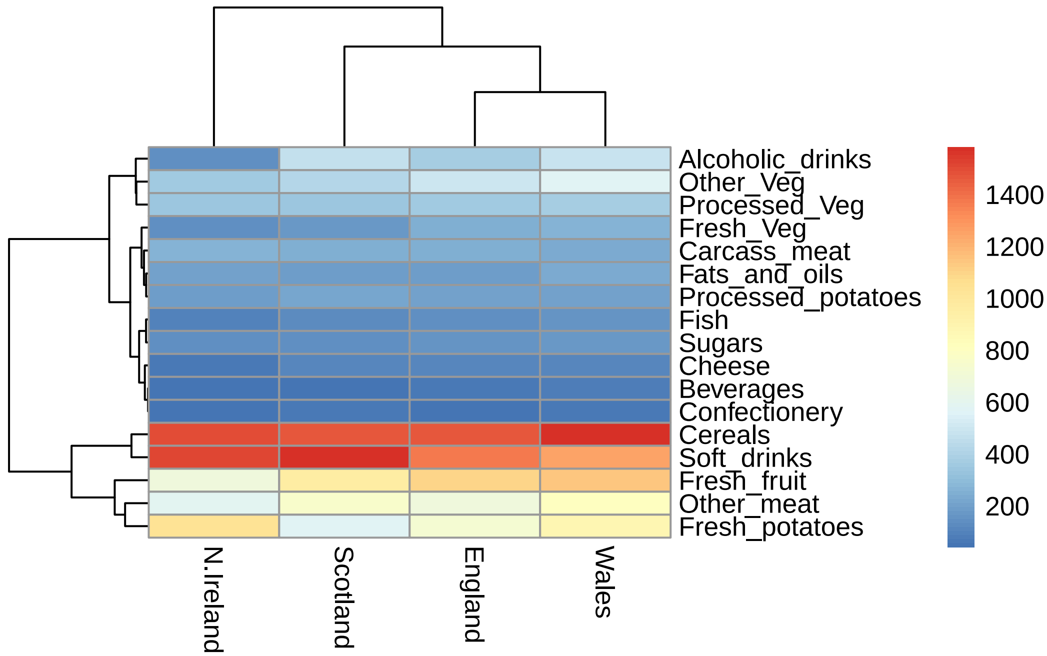
```
pairs(x, col=rainbow(nrow(x)), pch=16)
```



Heatmap

we can install the **pheatmap** package with the `install.packages()` commmand that we used perviously. Remember that we always run this in console 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? - The darker colors are higher values and the left is how food clustes together and the top is the countrys clustering togther

Of all of these plots really only `pairs()` plot was useful. tyhis however took a bit off work to interpret and well of scale when i am looking at much bigger datasets.

PCA the rescue

The main function in 'base R'bfor 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 varance is captured in the first Pc? - 67.4%
>
> Q. How many PC's do i need to capture to at least 90% of the total varance in the dataset? - 2 Pcs would capture 96.5% of total variance
>
> Q. Plot our main PCA result. Folks camnn call this diffrent things depending on their feild of study e. g. "PC plot", "ordienation plot", "score plot", "PC1 vs PC2" …

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

To generate our PCA score plot we want the `pca$x` componet of the resulting object

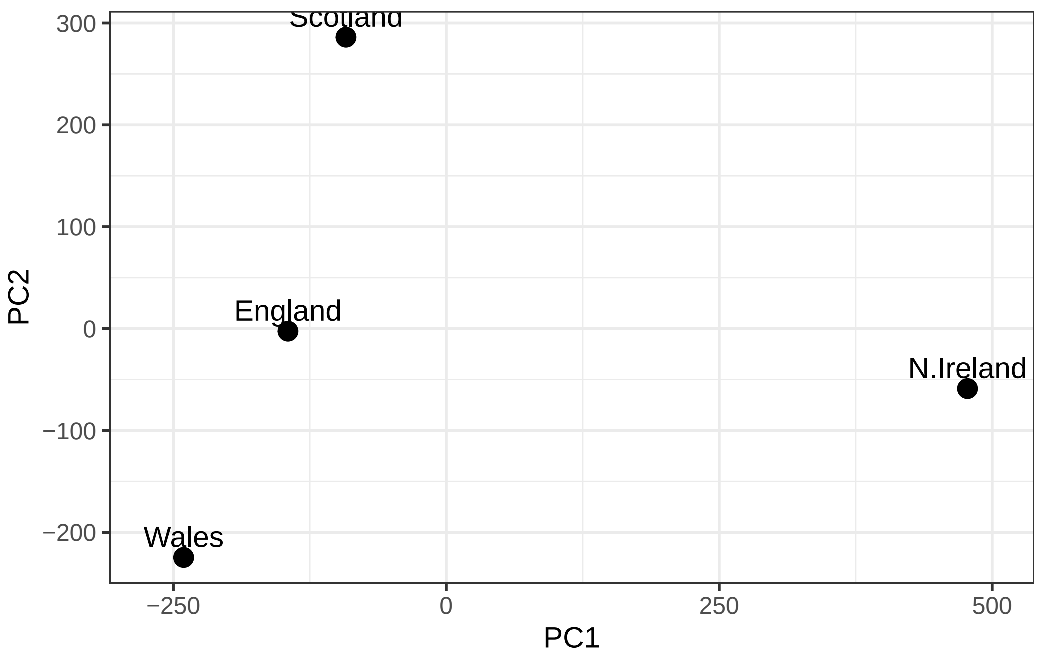
```
pca$x
```

```
                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
```

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

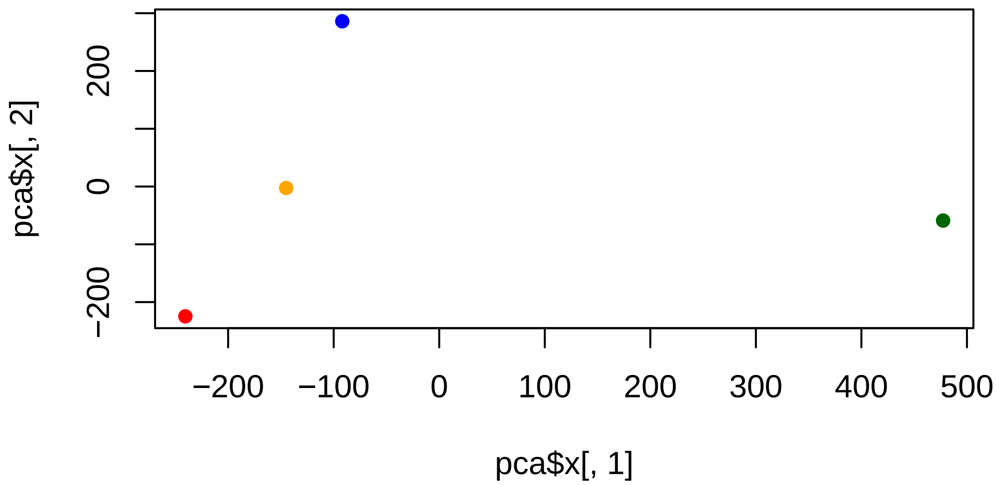
```
library(ggplot2)

# Create a data frame for plotting
df <- as.data.frame(pca$x)
df$Country <- rownames(df)

# Plot PC1 vs PC2 with ggplot
ggplot(pca$x) +
  aes(x = PC1, y = PC2, label = rownames(pca$x)) +
  geom_point(size = 3) +
  geom_text(vjust = -0.5) +
```

```
  xlim(-270, 500) +
  xlab("PC1") +
  ylab("PC2") +
  theme_bw()
```

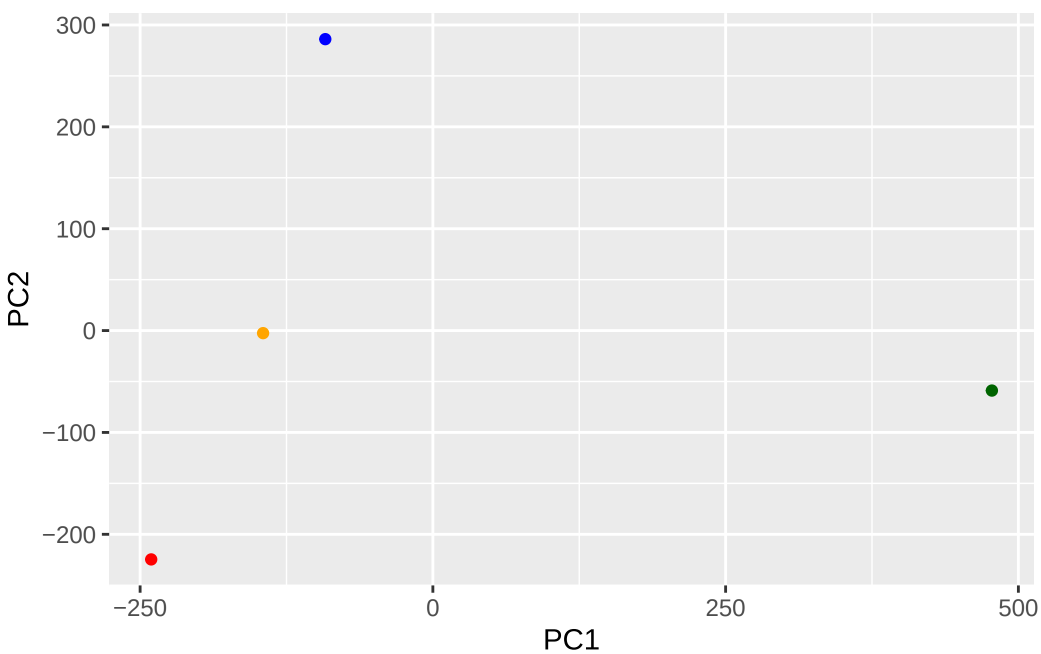


Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
my_cols <- c("orange", "red","blue", "darkgreen")
plot(pca$x[,1], pca$x[,2], col=my_cols, pch=16)
```




```
ggplot(pca$x) + aes(PC1, PC2) + geom_point(col=my_cols)
```



Digging Deeper (variable loadings)

How do the original variables (i.e. the 17 different foods contribute to our new PC's)

```
## Lets focus on PC1 as it accounts for > 90% of variance
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))
```