Class 7 Machine Learning

Joshua Khalil A17784122

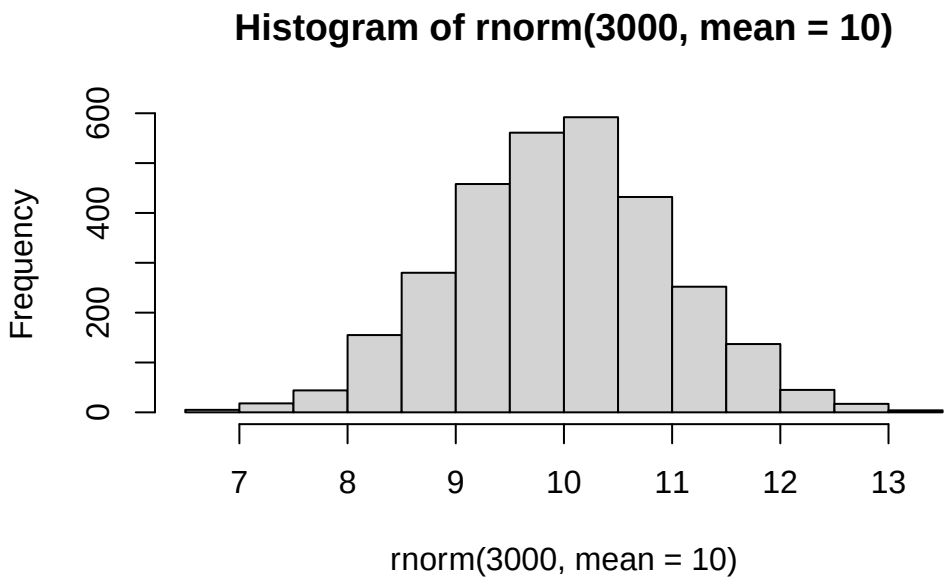
Table of contents

Background

Today we will begin exploration of important machine learning methods with a focus on how **clustering** and **dimensionallity reduction**.

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

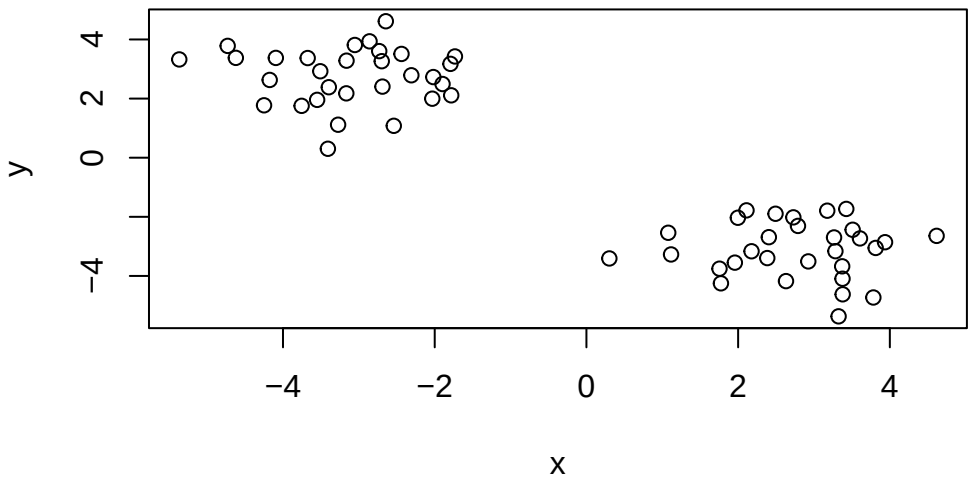
```
hist(rnorm(3000, mean =10))
```



Q. Can you generate 30 numbers centered at +3 and 30 numbers at -3 taken at 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 in "base R" for K-means clustering is called `kmeans()`

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

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

Cluster means:
          x         y
1 -3.111477  2.749367
2  2.749367 -3.111477

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 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] 52.8954 52.8954
 (between_SS / total_SS =  90.7 %)
```

```
Available components:

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

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

```
k$centers
```

```
          x         y
1 -3.111477  2.749367
2  2.749367 -3.111477
```

Q. What component of your kmeans result object has the cluster size (i.e. how many points are in each cluster)?

```
k$size
```

```
[1] 30 30
```

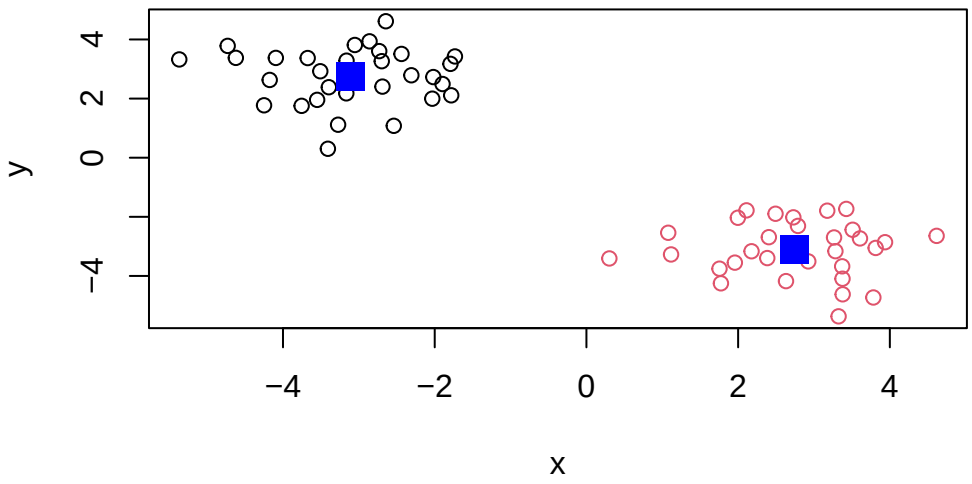
Q. What component of your kmeans result object has the cluster membership vector (i.e. the main clustering result: which points are in which cluster?

```
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 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 our data colored by clusterinf result) along with the cluster centers

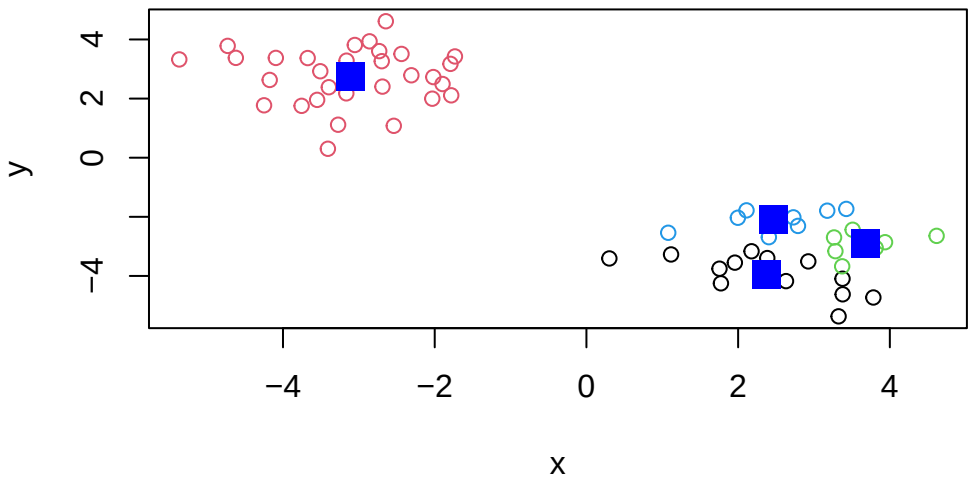
```
plot(x, col=k$cluster)
points(k$centers, col="blue", pch=15, cex=2)
```



Q. Can you runs `kmeans()` again and cluster 'x' into four 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)

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



Key-Point Kmeans will always return the clustering that we ask for (this is the "K" or "centers" in the K-means)!

```
k$tot.withinss
```

```
[1] 105.7908
```

Hierarchial Clustering

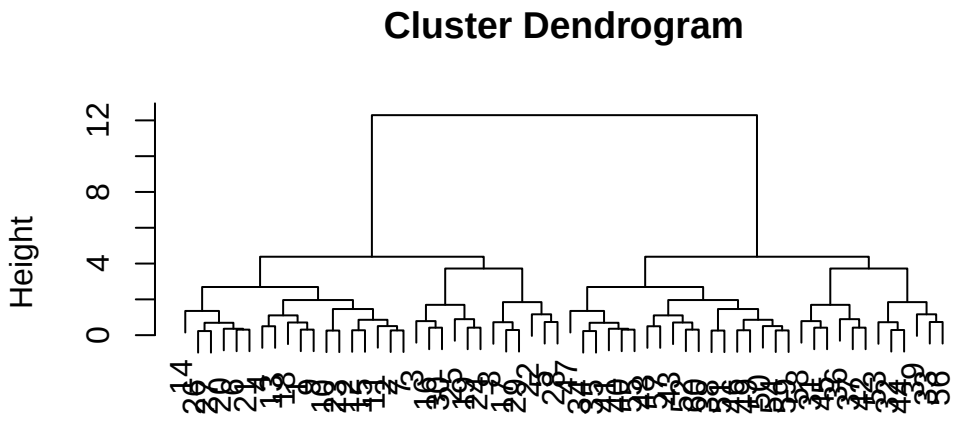
The main function for Hierarchical clustering in base R is called `hclust()`. One of the main differences with respect to the `kmeans()` function is that you can not pass your input data directly to `hclust` it needs a distance matrix. We can get this from lot's of places including the `dist` function.

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

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

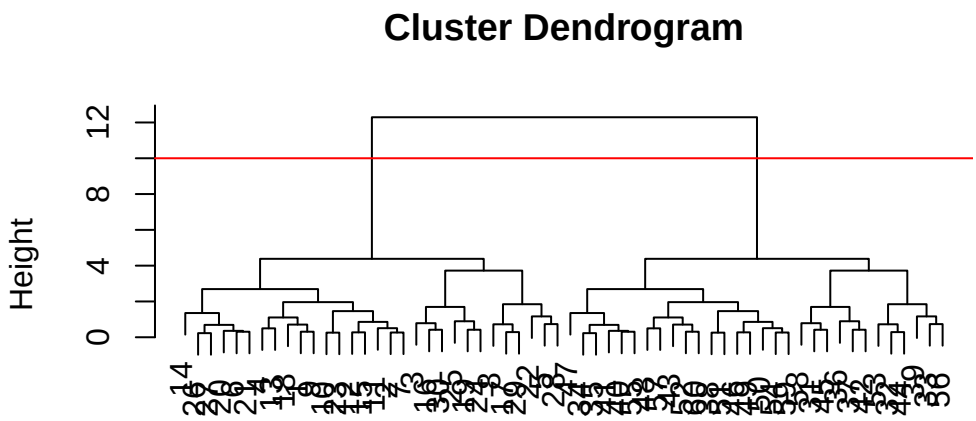
```
Cluster method   : complete
Distance         : euclidean
Number of objects: 60
```

```
plot(hc)
```



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

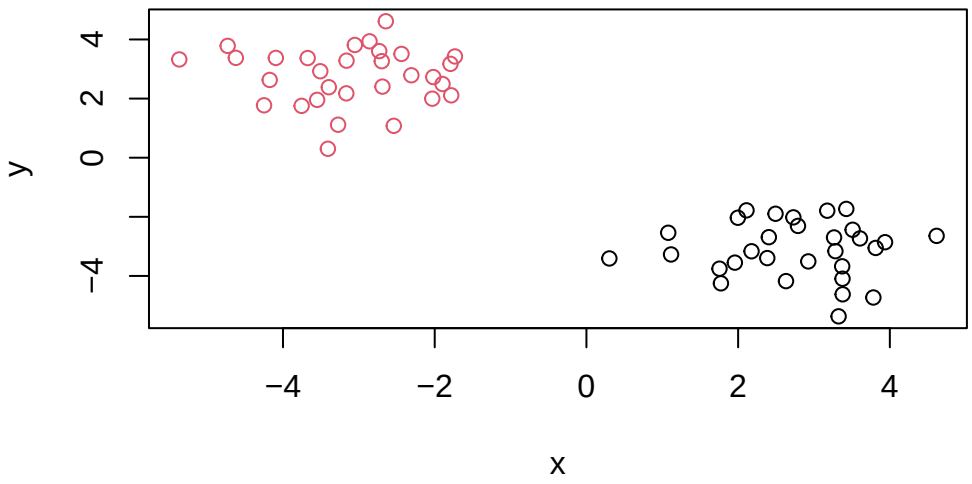
```
plot (hc)
abline(h=10, col="red")
```




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

Q. Plot our data `x` colored by the clustering results from `hclust()`

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



Principal Component Analysis (PCA)

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

PCA of UK food data

Read data on food consumption in UK

```
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 up 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 fix the row name assignment at import time

```
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?

```
dim(x)
```

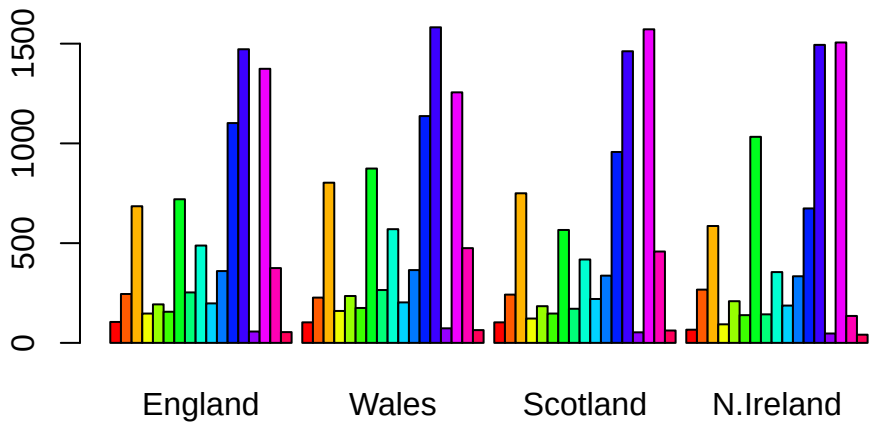
```
[1] 17  4
```

There are 17 variables and 4 countries

> 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?

The second approach is much more efficient, and the second is much more robust

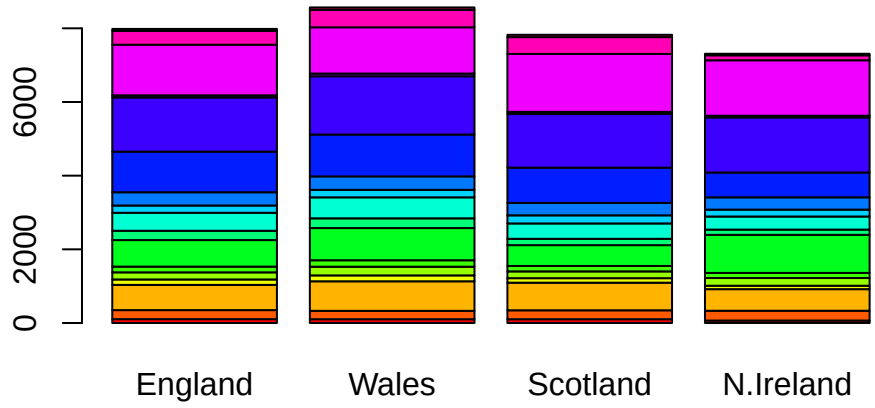
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

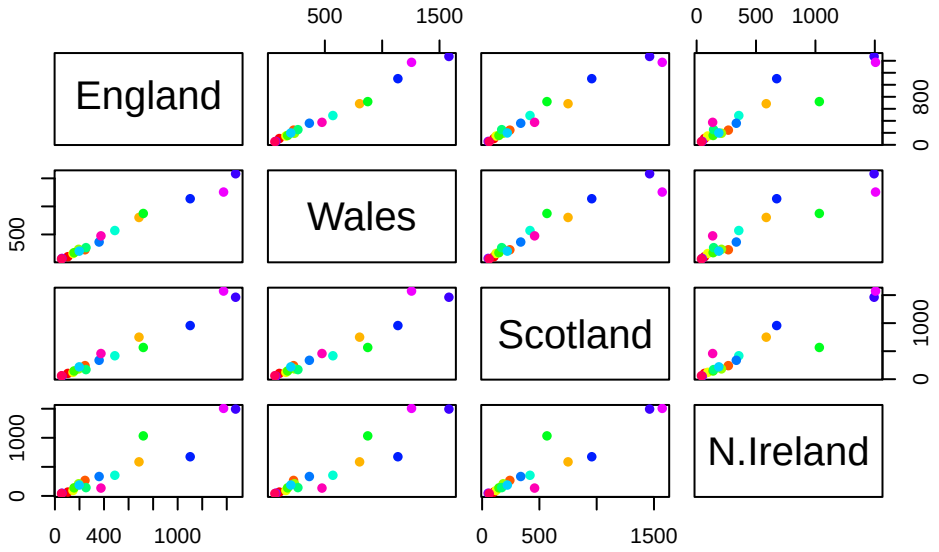
Changing the beside to false, as it is removing the idea of laying out the data beside each other, which would have the same effect if you were to leave the argument out.

```
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)
```



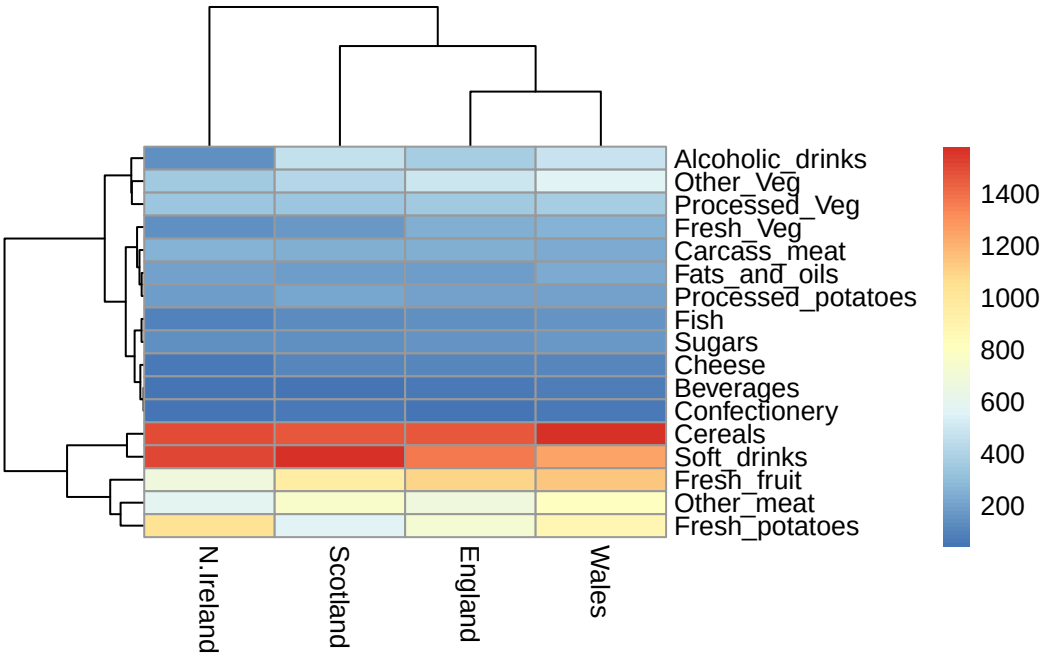
If a point lies on the diagonal, the two countries have the same value for the category being analyzed.

Heatmap

We can install the **pheatmap** package with the `install.packages()` command we used previously. Remember that we always run this in console…

```
library(pheatmap)

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



Of all the plots really only the `pairs` plot was useful. It took more to interpret

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?

67.4%

Q. How many PCs do I need to capture at least 90% of the total variance in the dataset?

2 PCs to capture 96.5% of the total variance.

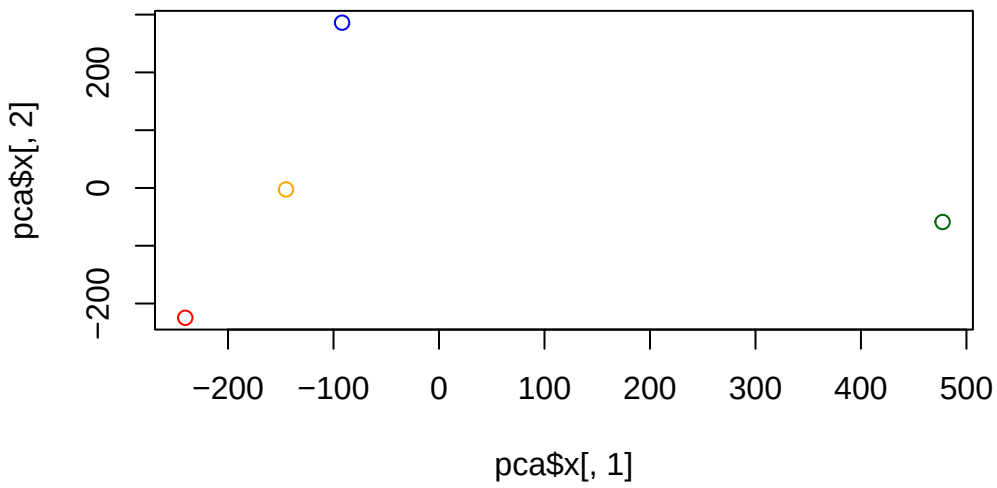
Q. Plot our main PCA result. ” PC Plot” or “Ordienation Plot” or “Score Plot”

to generate our PCA score plot we want the `pca$x` component of the result object

```
pca$x
```

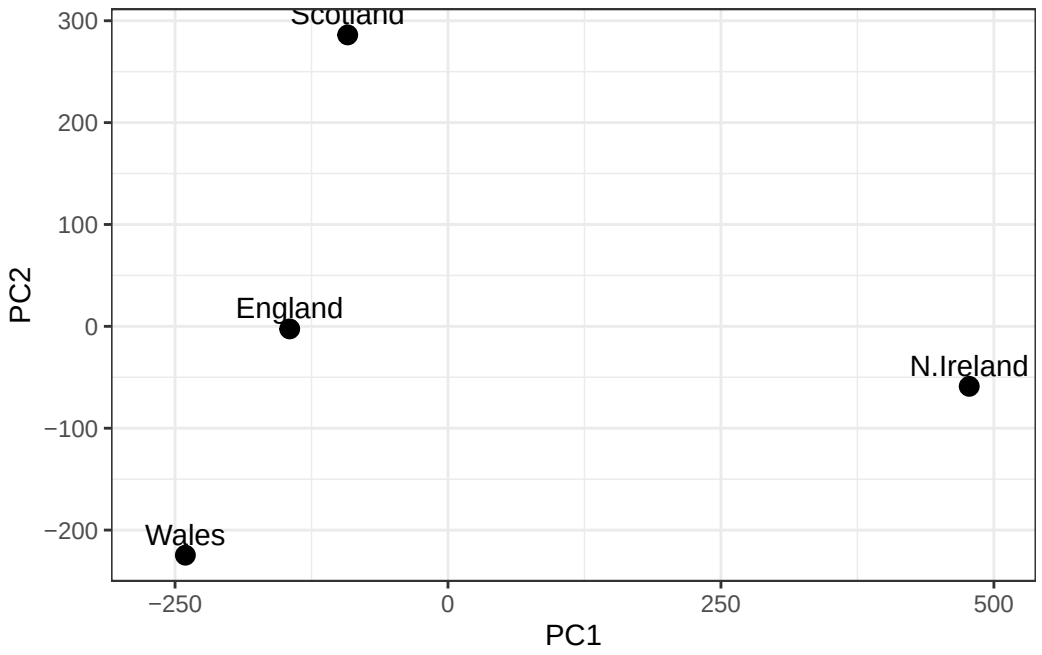
```
                PC1         PC2        PC3           PC4
England    -144.99315   -2.532999 105.768945  1.612425e-14
Wales      -240.52915 -224.646925 -56.475555  4.751043e-13
Scotland    -91.86934  286.081786 -44.415495 -6.044349e-13
N.Ireland   477.39164  -58.901862  -4.877895  1.145386e-13
```

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




```
library(ggplot2)
df <- as.data.frame(pca$x)
df$Country <- rownames(df)
ggplot(pca$x) +
  aes(x = PC1, y =PC2, label = rownames(pca$x)) +
  geom_point(size = 3) +
  geom_text(vjust = -0.5) +
```

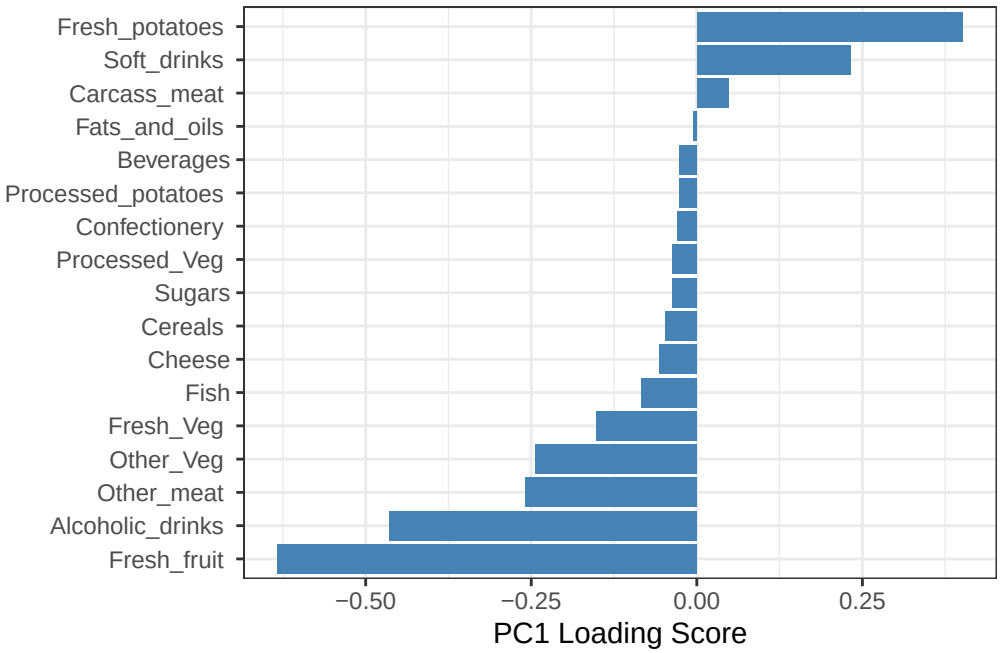
```
  xlim(-270, 500) +
  xlab("PC1") +
  ylab("PC2") +
  theme_bw()
```



Digging deeper (variable loadings)

How do the original variables (i.e. the 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