# Class 07: Machine Learning 1

Diego Diaz, PID: A17328629

## Table of contents

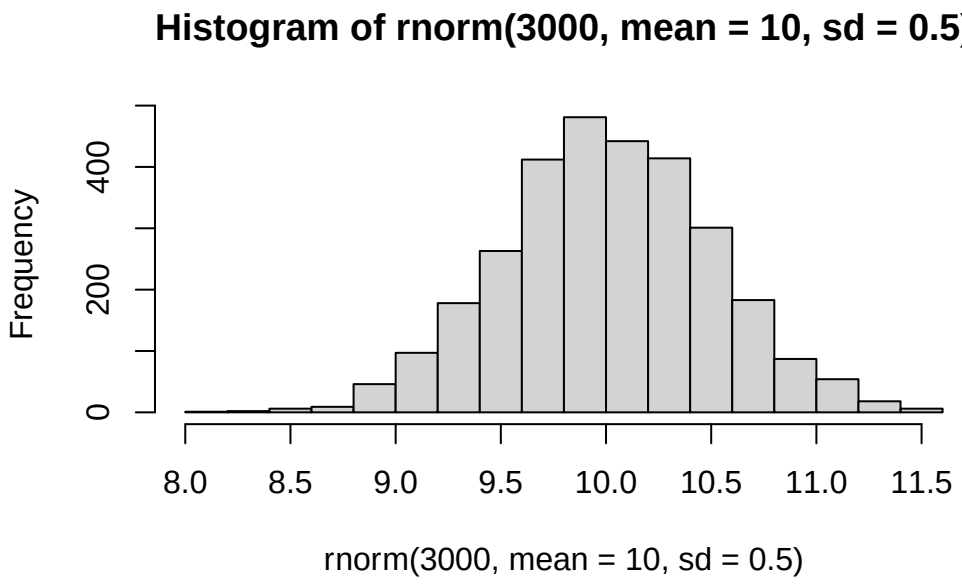
Background . . . . . 1
K-means clustering . . . . . 3
Hierachical clustering . . . . . 6
Principal Component Analysis (PCA) . . . . . 9
PCA of UK food data . . . . . 9
Heat Map . . . . . 14
PCA to the rescue . . . . . 15
Digging Deeper (variable loadings) . . . . . 17

### Background

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

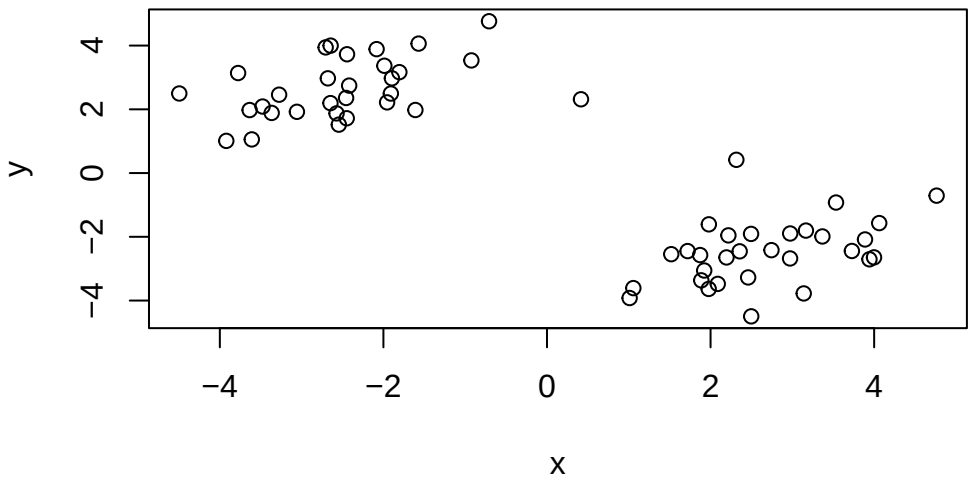
To start testing these methods let's make up some sample data to cluster where we know what the answer should be.

```
hist(rnorm(3000, mean=10, sd=0.5))
```




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

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




## K-means clustering

The main function in "base R" for k-means clustering is called `kmeans()`, let's try it out.

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

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

Cluster means:
          x         y
1  2.661457 -2.473678
2 -2.473678  2.661457

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

Within cluster sum of squares by cluster:
[1] 56.31699 56.31699
 (between_SS / total_SS =  87.5 %)
```

```
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  2.661457 -2.473678
2 -2.473678  2.661457
```

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
```

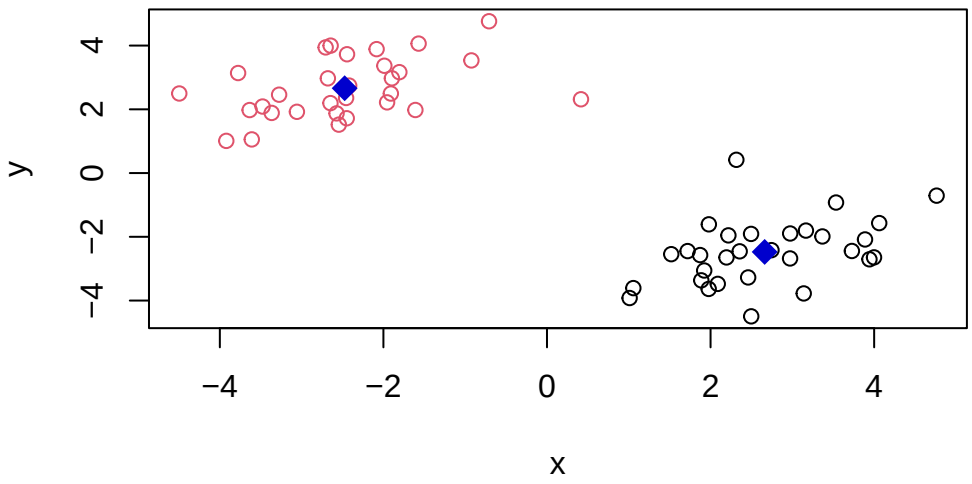
Q. What component of your kmeans result object has the cluster membership vector (i.e. the main clustering result: which points are in each vector)?

```
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 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
```

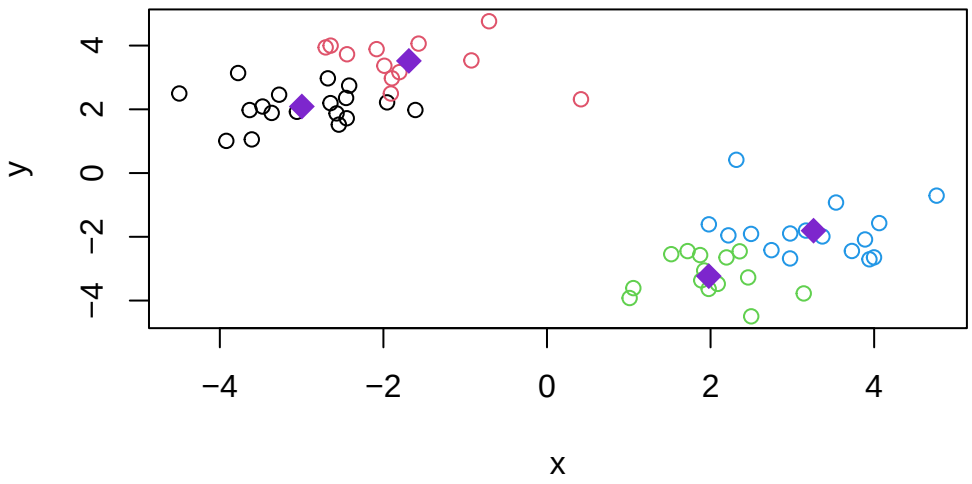
Q. Plot the results of clustering (i.e. means )

```
plot(x, col=c(k$cluster))
points(k$centers, col="blue3", pch=18, cex=1.8)
```




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

```
k2 <- kmeans(x, centers=4)
plot(x, col=c(k2$cluster))
points(k2$centers, col="purple3", pch=18, cex=1.8)
```




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

```
k$tot.withinss
```

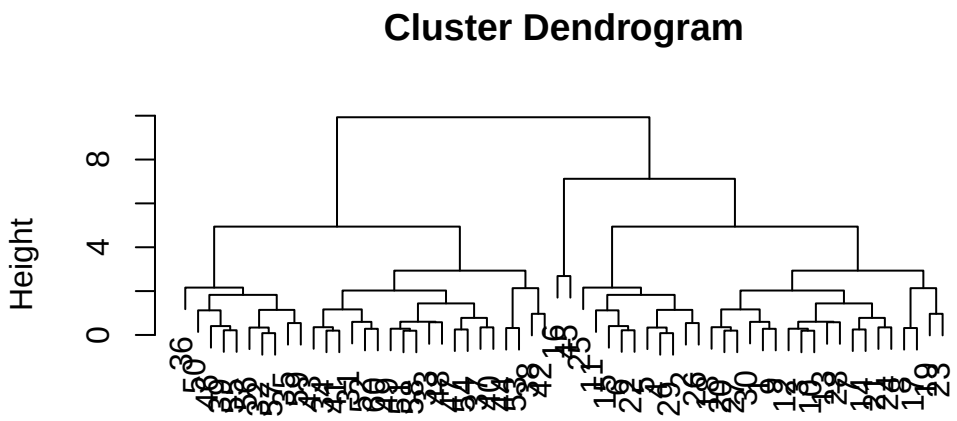
```
[1] 112.634
```

### Hierachical clustering

The main function for hierarchical clustering in base R is called `hclust()`.

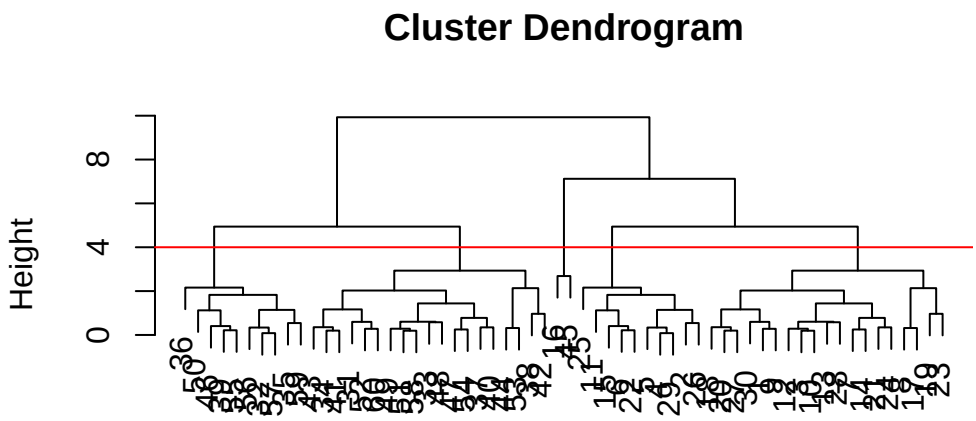
One of the main differences with respect to the `kmeans()` function is that you can not just pass your input data directly to `hclust()` - it needs a "distance matrix' as input. We can get this from lots of places including the `dist()` function.

```
xdist <- dist(x, diag=T, upper=T)
xhc <- hclust(xdist)
plot(xhc)
```




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

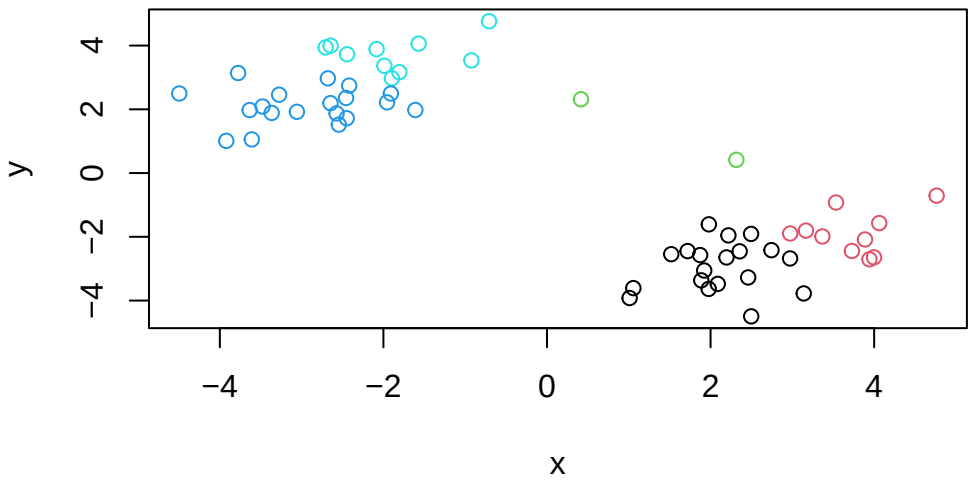
```
plot(xhc)
abline(h=4, col="red")
```




```
xmemb <- cutree(xhc, h=4)
```

Q. Plot our 'x' colored by the clusterinv result from `hclust()` and `cutree()`.

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




## Principal Component Analysis (PCA)

PCA is a popular dimensionallity reduction technique that's widely used in bioinformatics.

## PCA of UK food data

Start of lab sheet...

```
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?

```
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
```

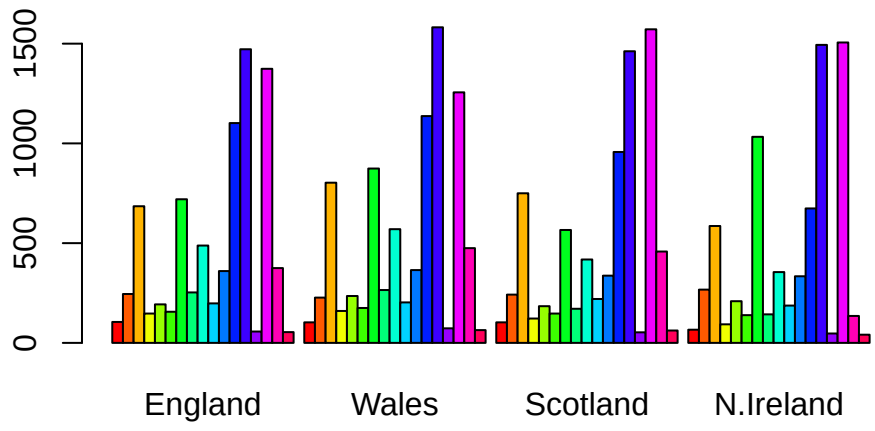
| | | | | |
|---|---|---|---|---|
| Other_meat | 685 | 803 | 750 | 586 |
| Fish | 147 | 160 | 122 | 93 |
| Fats_and_oils | 193 | 235 | 184 | 209 |
| Sugars | 156 | 175 | 147 | 139 |

There are 17 rows and 4 columns.

> 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?

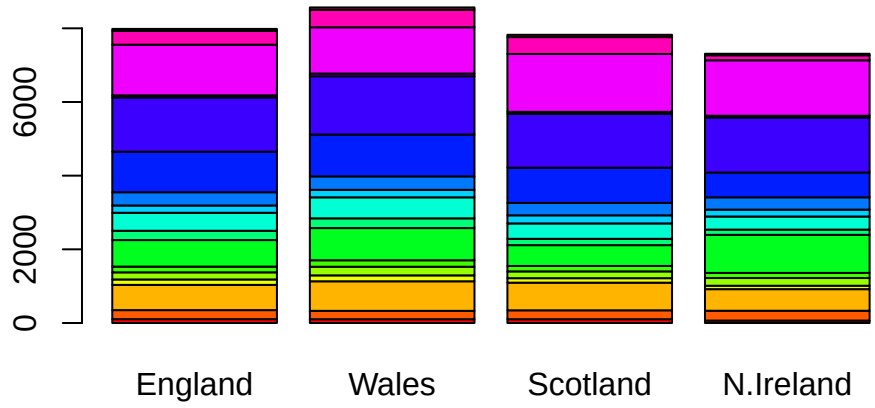
I prefer the `x <- read.csv(url, row.names=1)` method as it fixes the issue within the initial object itself rather than the problem being solved with subsequent lines of code, and I don't run the risk of deleting addiitional columns accidentally.

```
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=F, col=rainbow(nrow(x)))
```



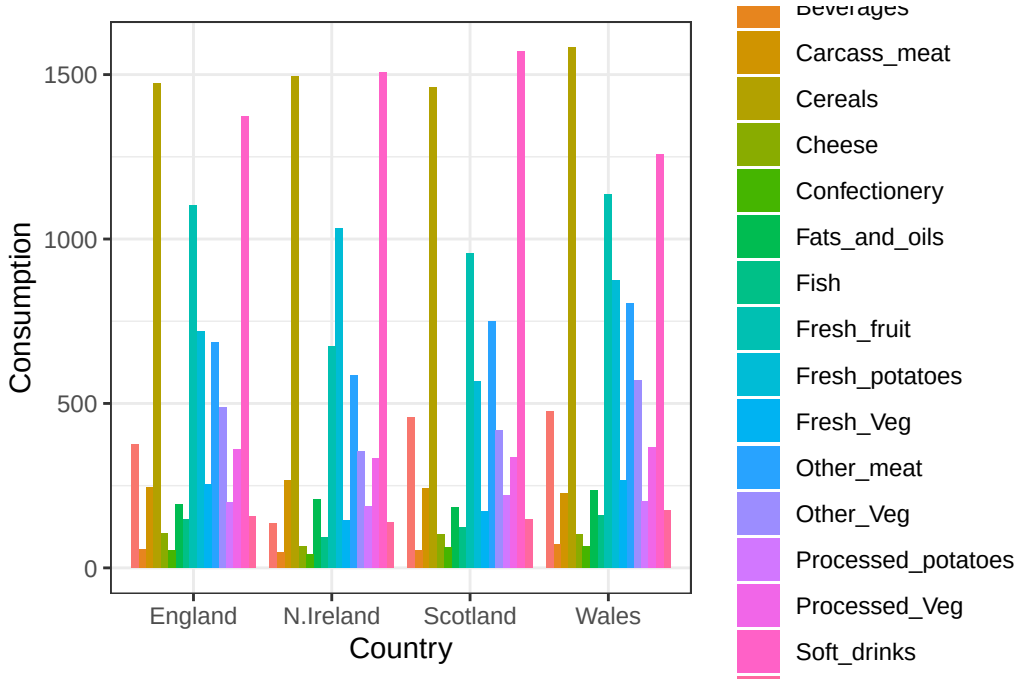

Converting data to "long" format (maximized rows and minimized columns) using `pivot_longer()` from the **tidyr** package.

```
library(tidyr)
x_long <- x |>
  tibble::rownames_to_column("Food") |>
  pivot_longer(cols=-Food,
               names_to = "Country",
               values_to = "Consumption")
dim(x_long)
```

```
[1] 68  3
```

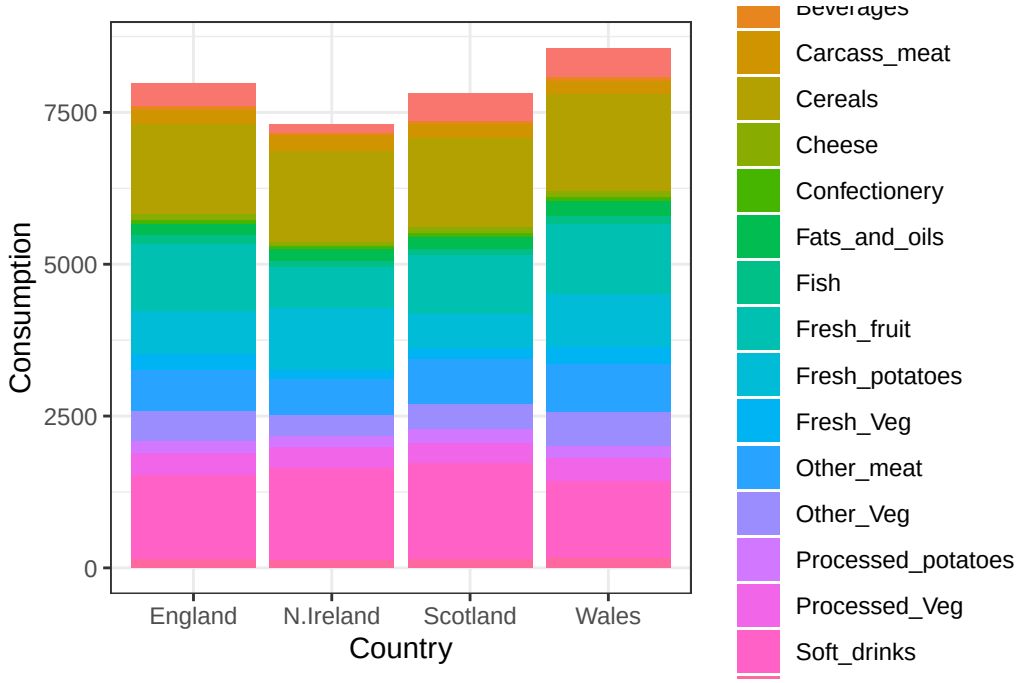
Creating a group bar plot using ggplot:

```
library(ggplot2)

ggplot(x_long) +
  aes(x=Country, y=Consumption, fill=Food) +
  geom_col(position="dodge") +
  theme_bw()
```



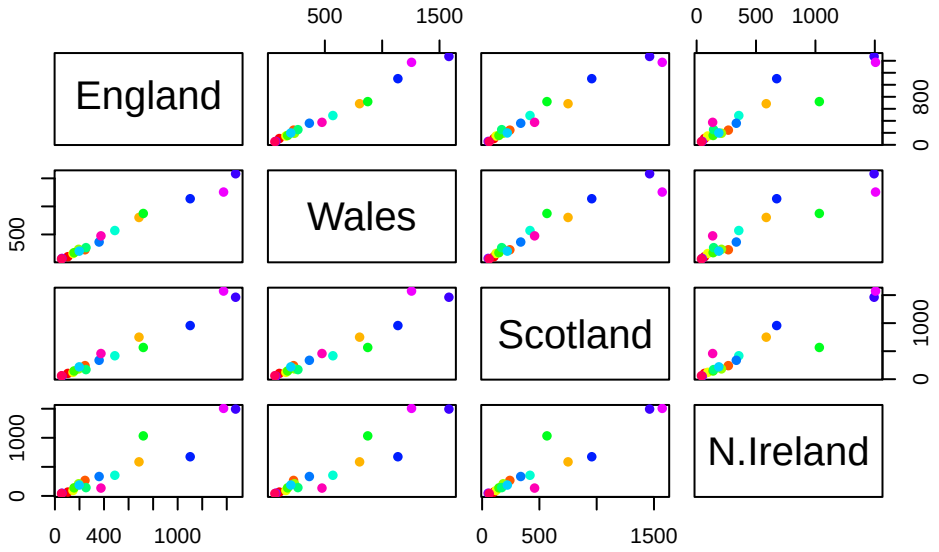

Q4: Changing what optional argument in the above ggplot() code results in a stacked barplot figure?

```
library(ggplot2)

ggplot(x_long) +
  aes(x=Country, y=Consumption, fill=Food) +
  geom_col(position="stack") +
  theme_bw()
```




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)
```

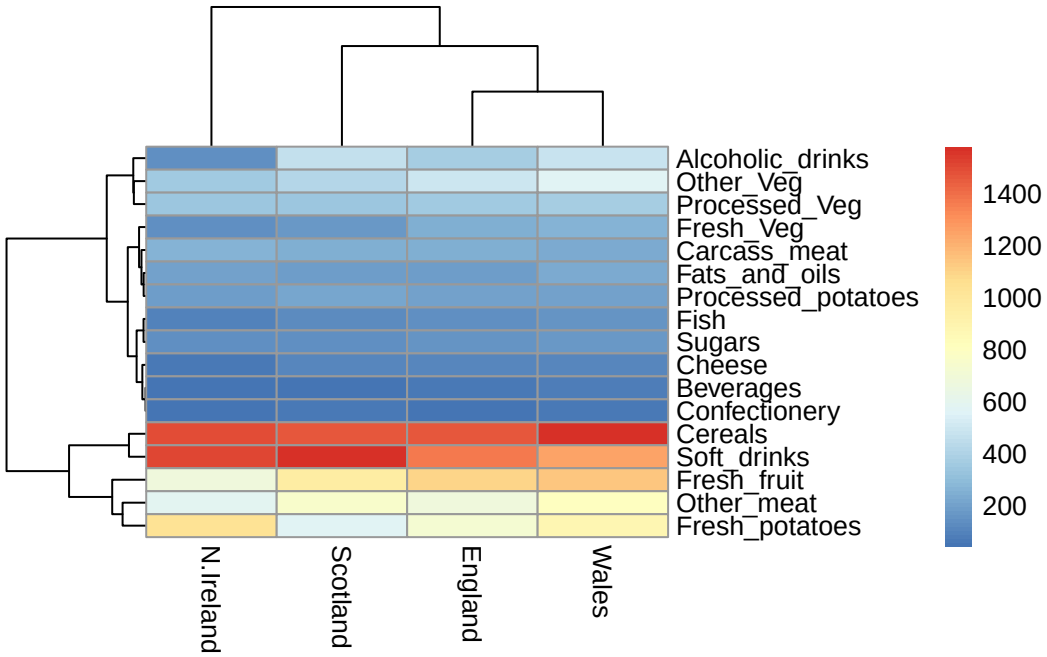



This function/plot shows the relationships between variables in the matrix. Points that lie on the diagonal mean show a correlation between the two given variables compared.

### Heat Map

We can install the **pheatmap** package with the `install.packages()` commmand that we used previously. Remember that we always run this command in the R consol and **not** the 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?

England, Wales, and Scotland cluster together, suggesting an similar pattern in food consumption between these countries. N. Ireland seems to diverge the most in alcoholic drink, fresh vegetable, and other meat consumption.

## 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 3.176e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

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

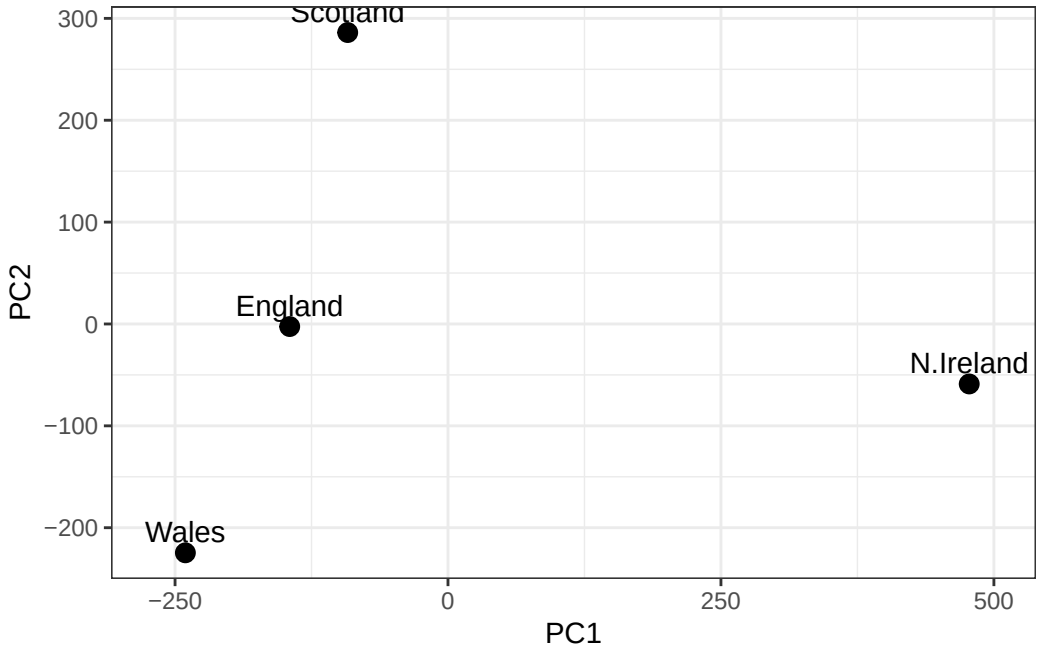
0.6744/67.44%

Q. How many PCs do I need captured?

Two PCs capture 96.5% of the total variance.

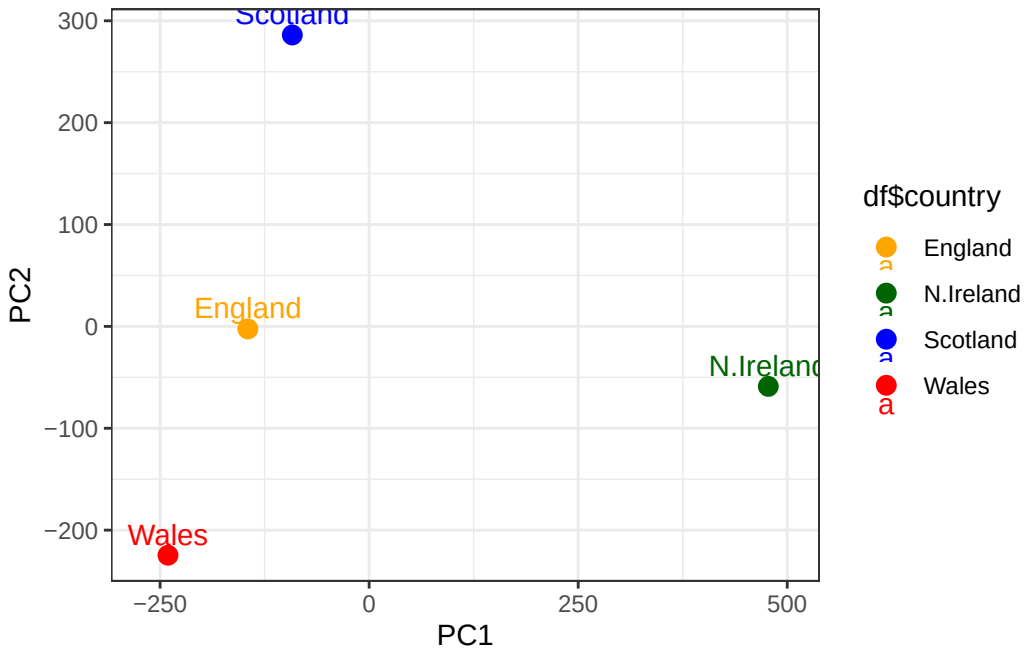
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
df <- as.data.frame(pca$x)
df$country <- rownames(df)
ggplot(pca$x) +
  aes(x=df$PC1,y=df$PC2, label=rownames(pca$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.

```
cols <- c("Wales" = "red", "England" = "orange", "Scotland" = "blue", "N.Ireland" = "darkgree
df <- as.data.frame(pca$x)
df$country <- rownames(df)
ggplot(pca$x) +
  aes(x=df$PC1,y=df$PC2, label=rownames(pca$x), col=df$country) +
  geom_point(size=3) +
  geom_text(vjust=-0.5)+
  scale_color_manual(values=cols) +
  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?

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

The two food groups that are featured prominantely are fresh potatoes (negative), and soft drinks (positive). PC2 mainly tells us countroes with a prominently processed food diet versus diets with more fresh components.

```
ggplot(pca$rotation) +
  aes(x = PC2,
      y = reorder(rownames(pca$rotation), PC2)) +
  geom_col(fill = "steelblue") +
  xlab("PC2 Loading Score") +
  ylab("") +
  theme_bw() +
  theme(axis.text.y = element_text(size = 9))
```