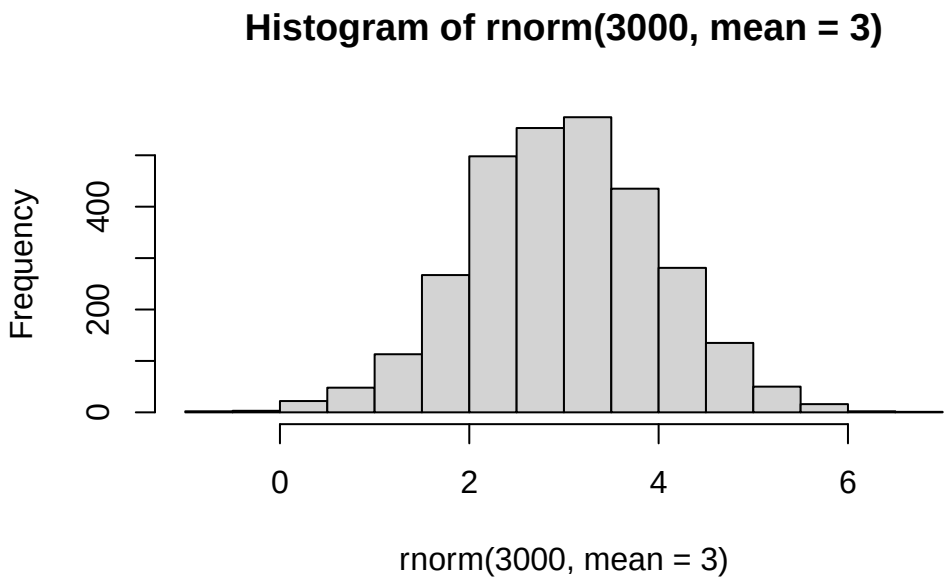
Class 7: Machine Learning

Sze Sze Chan (PID 167)

Today we will explore unsupervised machine learning methods including clustering and dimensionality reduction methods.

Let's start by making up some data (where we know there are clear groups) that we can use to test out different clustering methods.

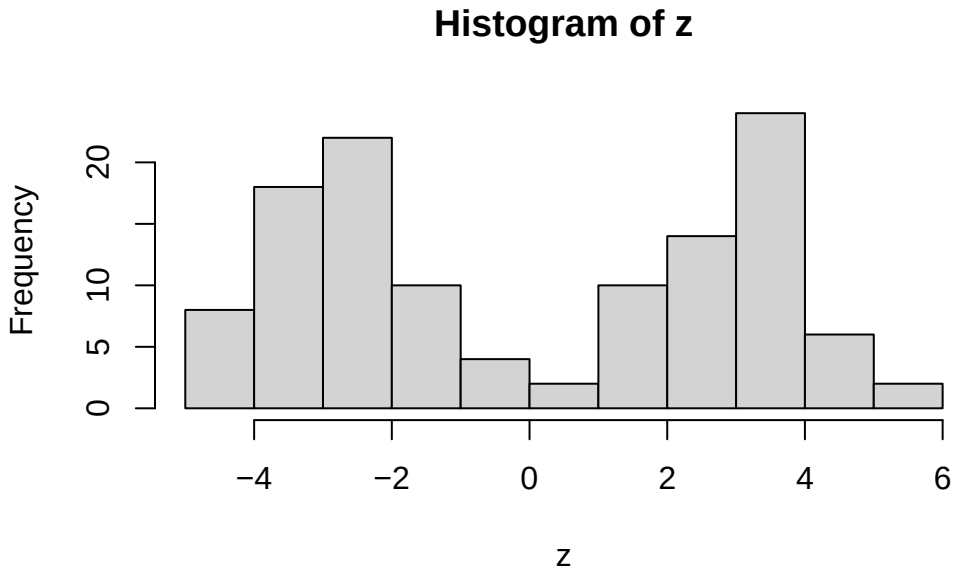
```
hist(rnorm(3000, mean=3))
```



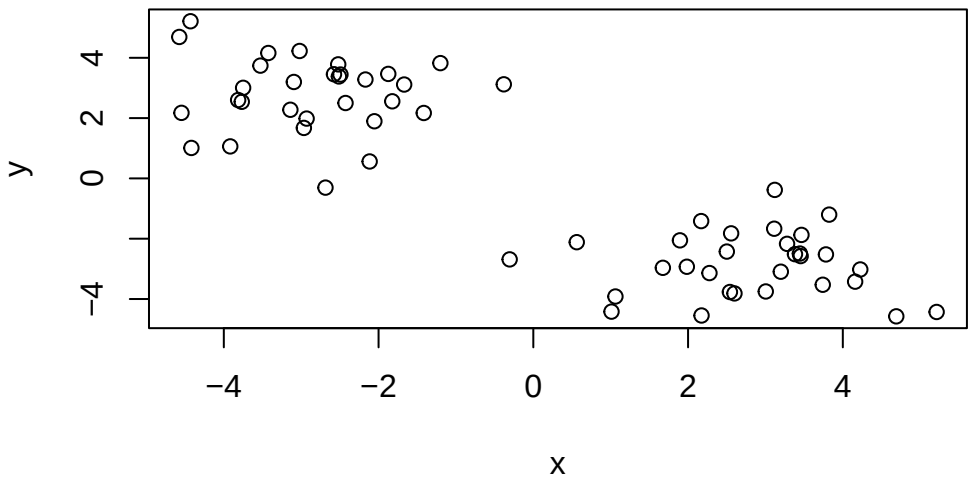
Make data with two "clusters"

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

z <-cbind(x=x, y=rev(x))
hist(z)
```




```r
plot(z)
```



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

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

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

Cluster means:
          x         y
1  2.793269 -2.841445
2 -2.841445  2.793269

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] 75.44139 75.44139
 (between_SS / total_SS =  86.3 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
```

```
[6] "betweenss"    "size"         "iter"         "ifault"
```

How big is 'z'

```
attributes(k)
```

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

$class
[1] "kmeans"
```

> Q. How many points lie in each cluster?

```
k$size
```

```
[1] 30 30
```

> Q. What component of our results tells us about the cluster membership (ie which point likes 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
```

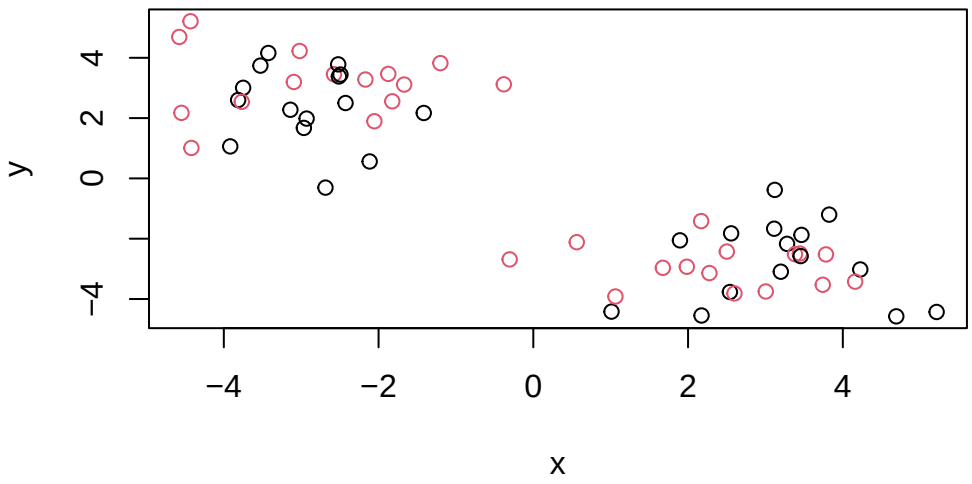
```
k$center
```

```
          x         y
1  2.793269 -2.841445
2 -2.841445  2.793269
```

> Q. Put this info togetherr and make a little "base R" plot of our clustering result. ALso add the cluster center points to this plot.

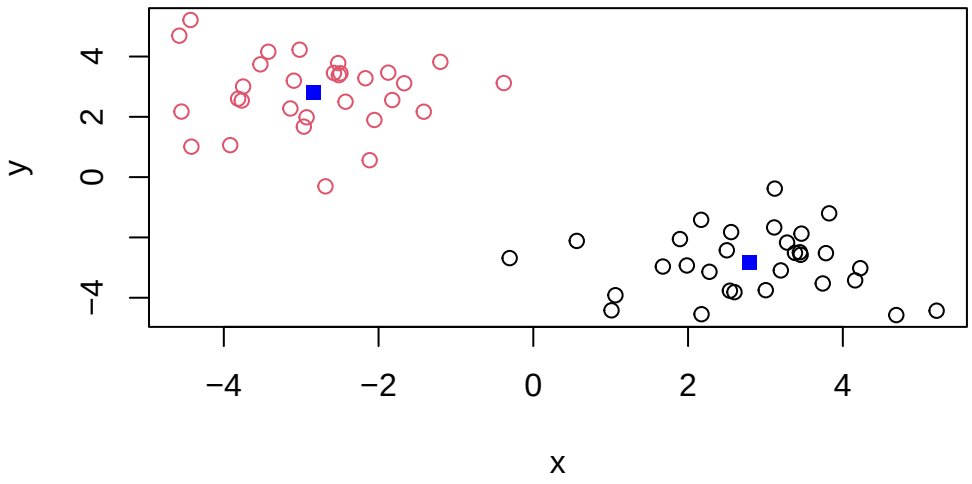
You can color by number (1st from the color palette)

```
plot(z, col=c(1,2))
```



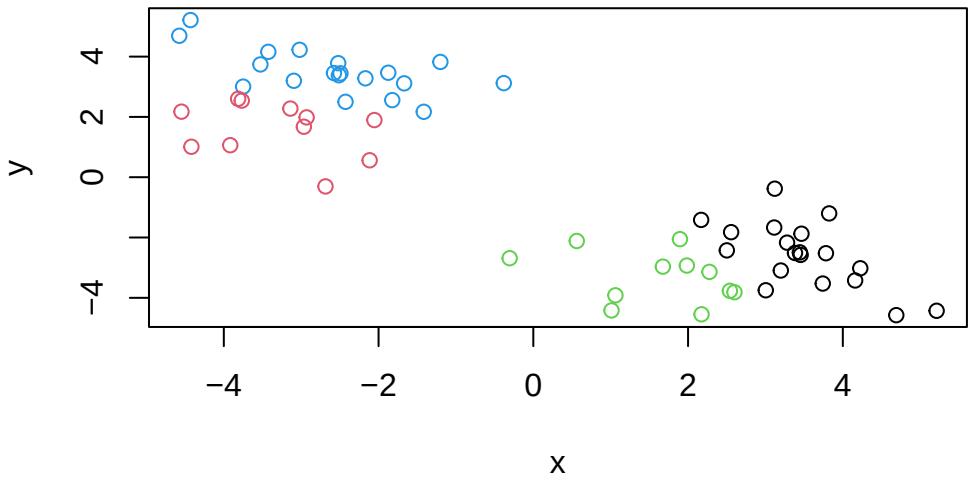
Plot colored by cluster membership:

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



Q. Run Kmeans on our input 'z' and define 4 clusters making the same result visualization plot as above (plot of z collared by cluster membership)

```
k4 <- kmeans(z, centers = 4)
plot (z, col=k4$cluster)
```



Hierarchial Clustering

The main function in base R for this called 'hclust()' it will tkae as input a distance matrix (key point is that yoy cant just give your "raw" data input - you have to first calculate a distance matrix from your data).

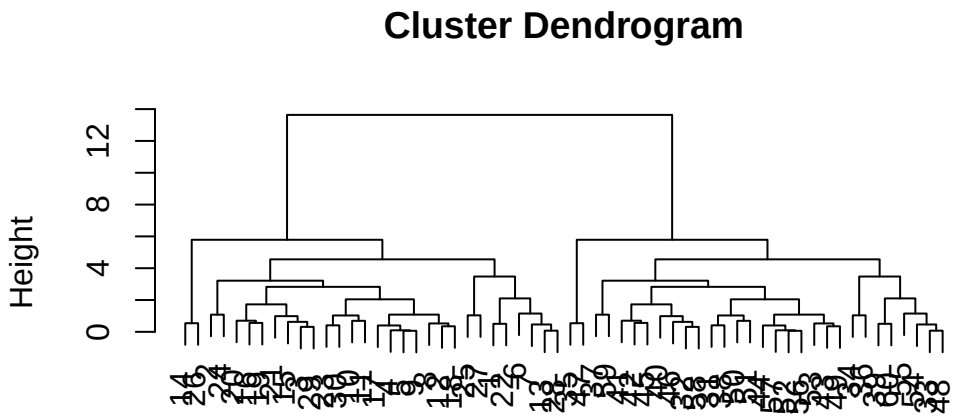
```
d <- dist(z)
hc <- hclust(d)
hc
```

```
Call:
hclust(d = d)

Cluster method   : complete
```

```
Distance         : euclidean
Number of objects: 60
```

```
plot(hc)
```

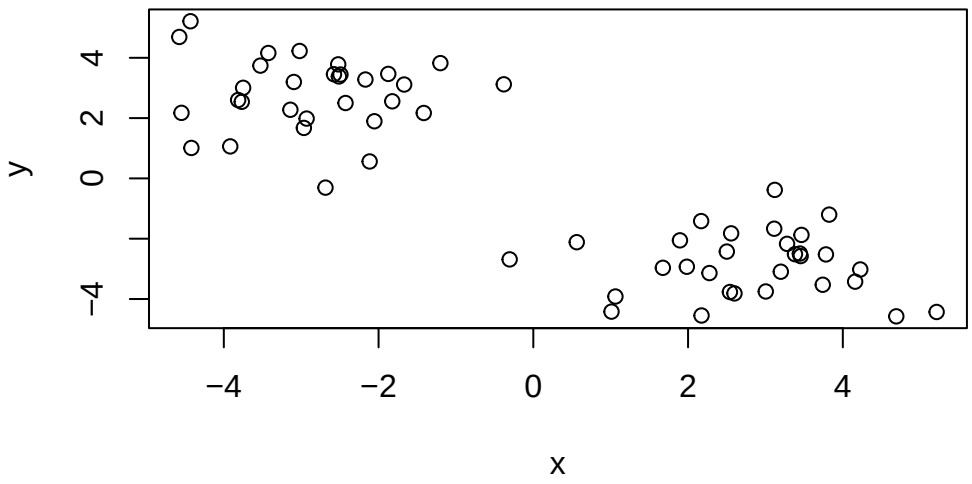


Once I inspect the “tree” I can “cut” the three to yeild my groupings or clusters. The functiuon to do this is called ’cutree()

```
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 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
```

```
plot(z)
```




```
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
```

###Data Input

```
dim(x); ncol(x); nrow(x)
```

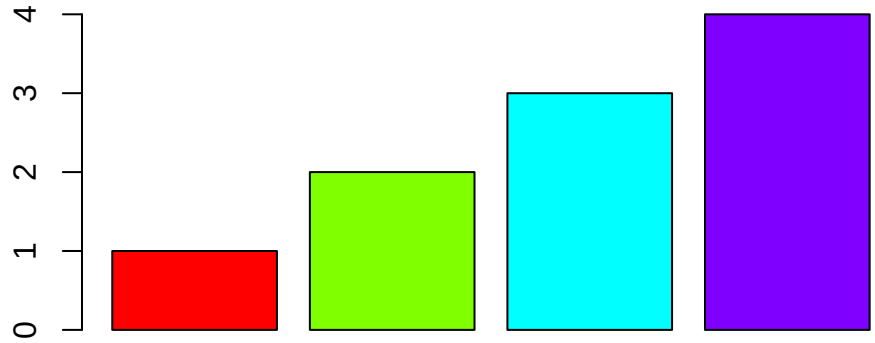
```
[1] 17  5
```

```
[1] 5
```

```
[1] 17
```

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

```
x <- c(1, 2, 3, 4)  # Example vector
barplot(x, col = rainbow(length(x)))
```



Looking at these types of pairwise plots can be helpful but it does not scale well and kind of sucks

###PCA to the rescue

Main functon for pca in base R is called 'prcomp()'. This function wants to transpose of our input data ie the importantnt foods in as columns and the countries as rows.

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

```
Importance of components:
                         PC1
Standard deviation         0
Proportion of Variance   NaN
Cumulative Proportion    NaN
```

Lets see what is in your pca result object pca

The 'pcas$' result object is where we focus on details how the countires are related to each othewr in temrs of our new "axis" *pcs, evigenvectrors etc"

We cam look at the so called pc loadings result object to see how (the background foods contribute to our new better variables)

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
```

```
# Note how the minus indexing works
rownames(x) <- x[,1]
x <- x[,-1]
head(x)
```

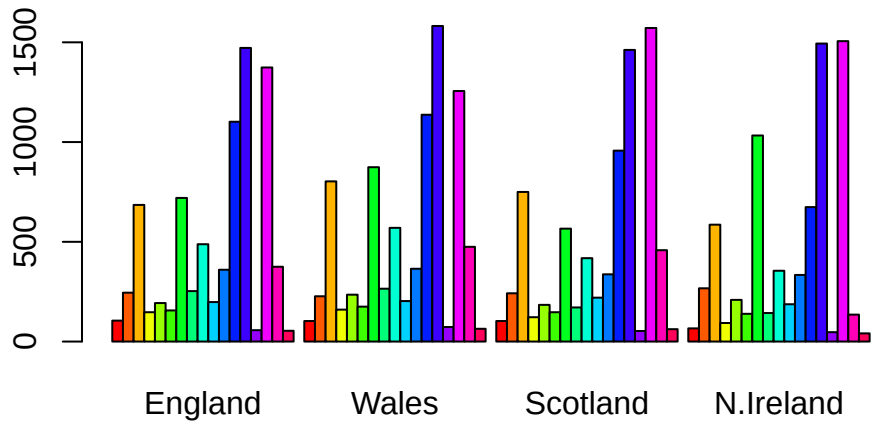
```
               England Wales Scotland N.Ireland
Cheese             105   103      103        66
Carcass_meat       245   227      242       267
Other_meat         685   803      750       586
Fish               147   160      122        93
Fats_and_oils      193   235      184       209
Sugars  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

```
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

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?

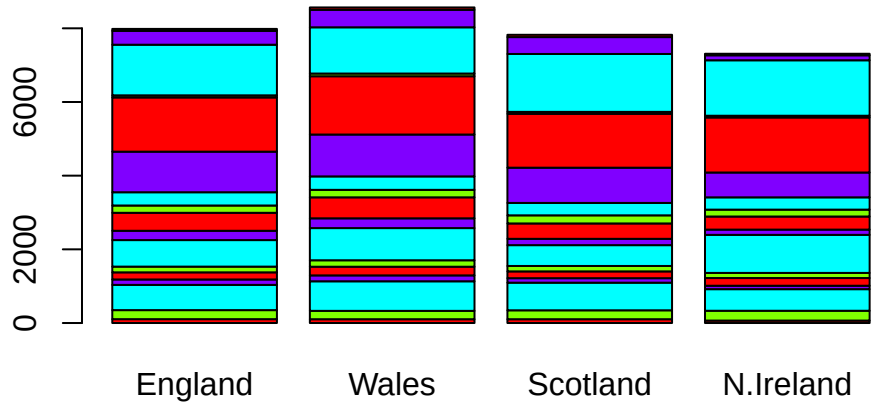
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



Add labels. It directly specifys row names as labels so it makes understanding easier.

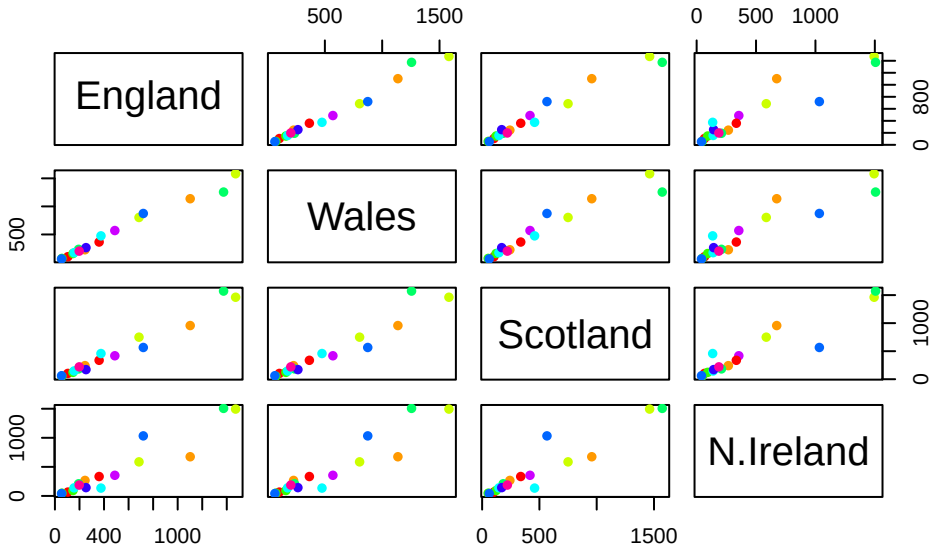
Q3. Changing what optional argument in the above barplot() function results in the following plot?

```
barplot(as.matrix(x), beside=FALSE, col=rainbow(ncol(x)), names.arg=colnames(x))
```



Q5: Generating all pairwise plots may help somewhat. 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(10), pch=16)
```



It is a pairwise scatter plot matrix. The graphs show the correlations between variable pairs and compares them. 2 variables have the same value if the position of a point is exactly at the diagonal.

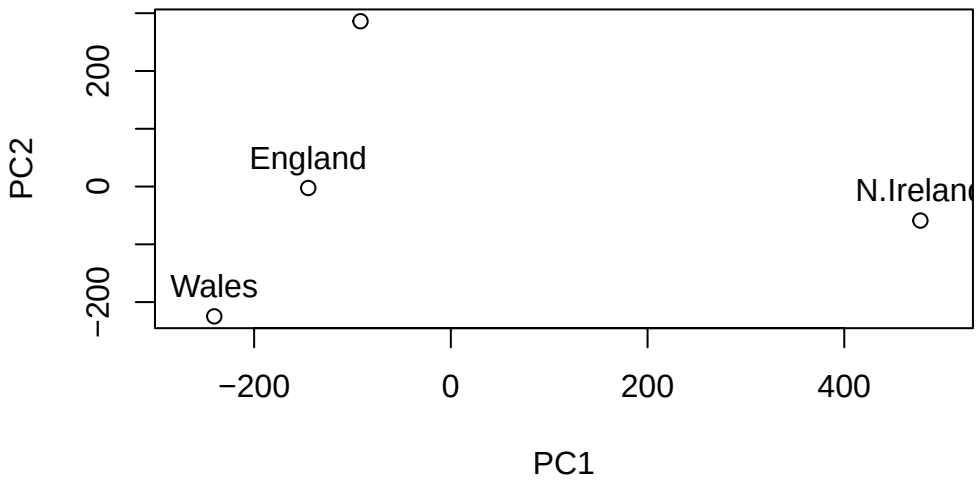
Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set?

```
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
```

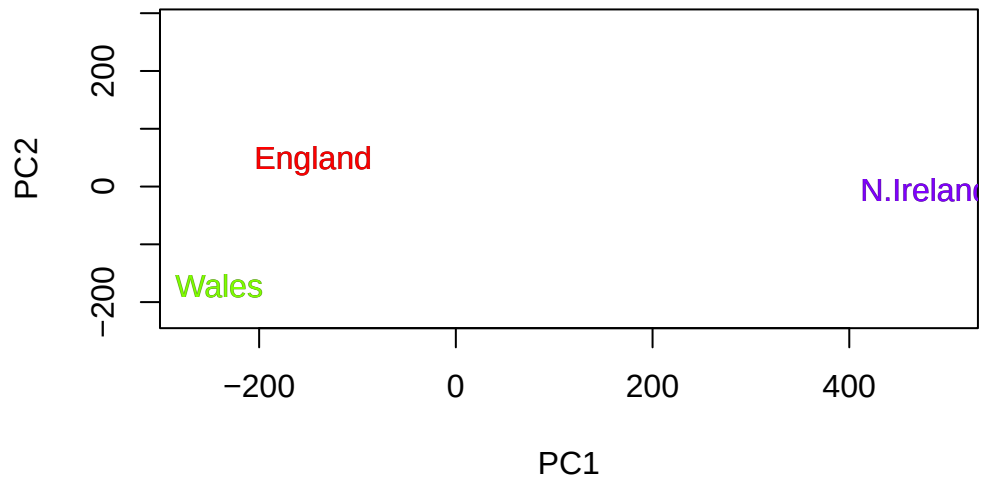
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
plot(pca$x[, 1], pca$x[, 2], xlab="PC1", ylab="PC2", xlim=c(-270, 500))
text(pca$x[, 1], pca$x[, 2], labels=colnames(x), pos=3)
```



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.

```
plot(pca$x[, 1], pca$x[, 2], xlab="PC1", ylab="PC2", xlim=c(-270, 500),
     col=NA)  # col=NA makes the points invisible
text(pca$x[, 1], pca$x[, 2], labels=colnames(x), pos=3)
colors <- rainbow(ncol(x))  # Generate a vector of colors
text(pca$x[, 1], pca$x[, 2], labels=colnames(x), pos=3, col=colors)
```



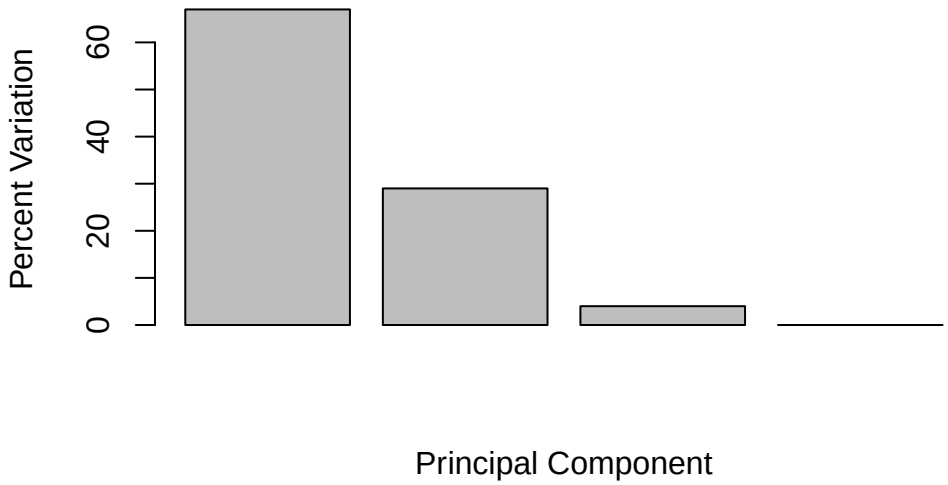

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

```
[1] 67 29  4  0
```

```
## or the second row here...
z <- summary(pca)
z$importance
```

```
                             PC1       PC2      PC3          PC4
Standard deviation     324.15019 212.74780 73.87622 3.175833e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

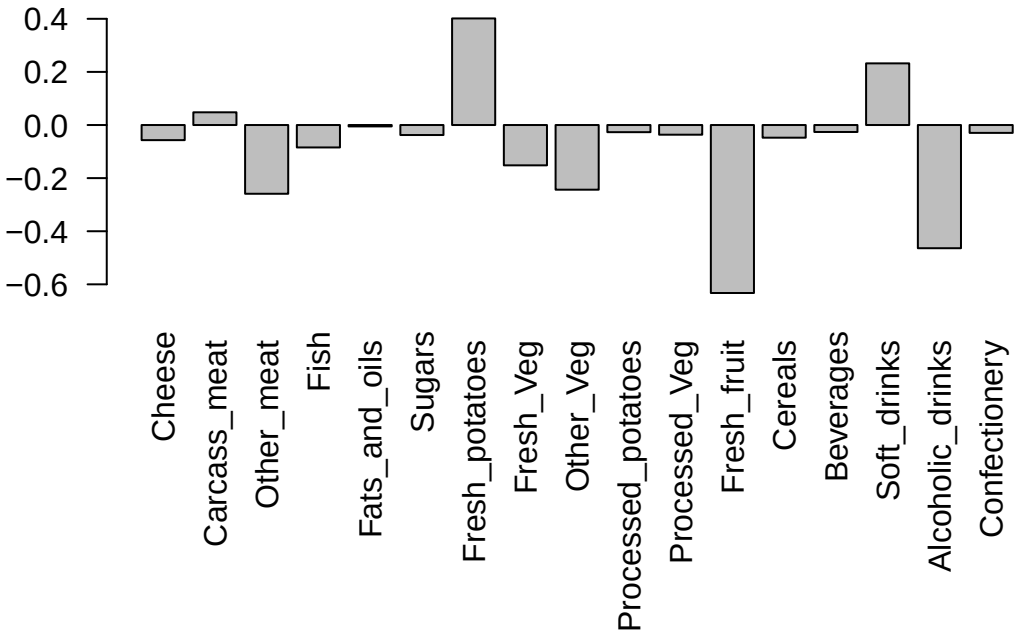
```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```



Lets focus on PC1 as it accounts for $>$ 90% of variance

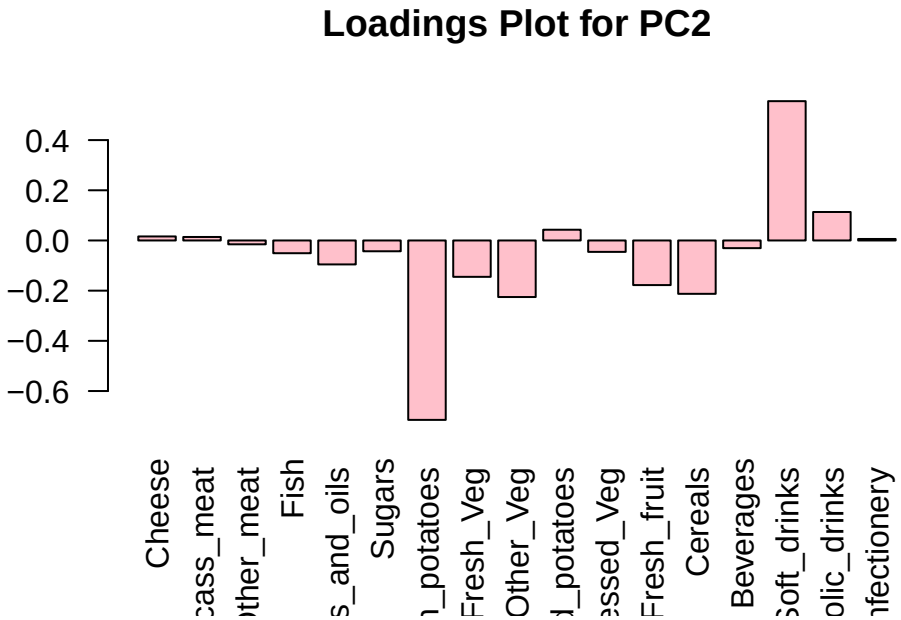
par(mar=c(10, 3, 0.35, 0)) barplot(pca$rotation[,1], las=2)

```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



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

```
barplot(pca$rotation[, 2], las=2, main="Loadings Plot for PC2", col = "pink")
```

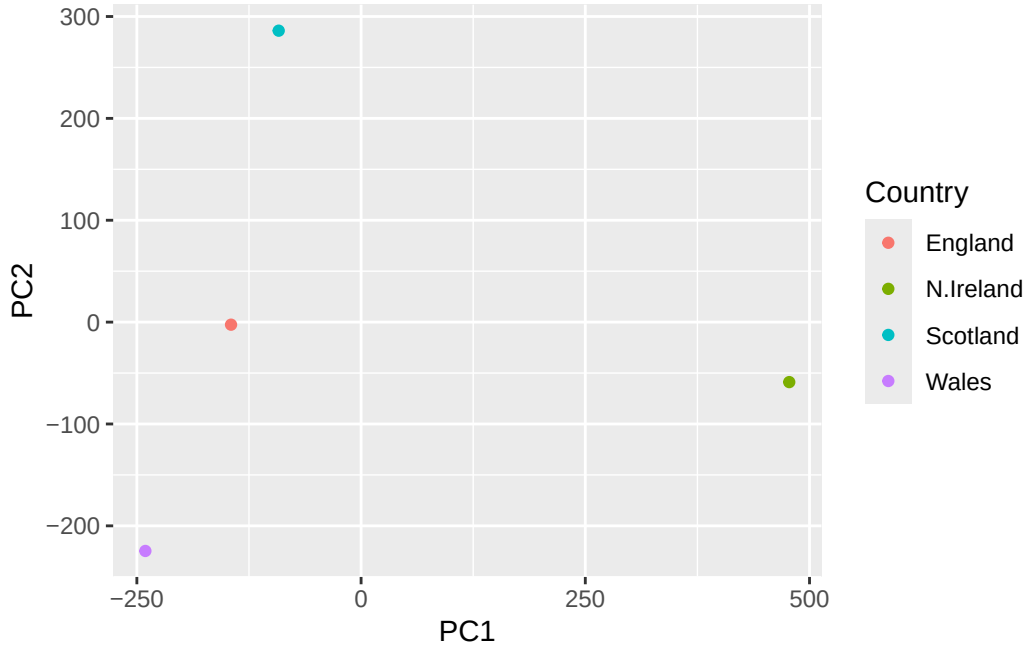


Feature prominantly fresh potatoes and soft drinks. Pc2 mainly tells us about the difference between north irelands diet and other countries' diet (there is more starch and sugar in North Ireland's diet).

```
library(ggplot2)

df <- as.data.frame(pca$x)
df_lab <- tibble::rownames_to_column(df, "Country")

# Our first basic plot
ggplot(df_lab) +
  aes(PC1, PC2, col=Country) +
  geom_point()
```




```
url2 <- "https://tinyurl.com/expression-CSV"
rna.data <- read.csv(url2, row.names=1)
head(rna.data)
```

```
       wt1 wt2  wt3  wt4 wt5 ko1 ko2 ko3 ko4 ko5
gene1  439 458  408  429 420  90  88  86  90  93
gene2  219 200  204  210 187 427 423 434 433 426
gene3 1006 989 1030 1017 973 252 237 238 226 210
```

```
gene4  783 792  829  856 760 849 856 835 885 894
gene5  181 249  204  244 225 277 305 272 270 279
gene6  460 502  491  491 493 612 594 577 618 638
```

Q10: How many genes and samples are in this data set? Genes: 6 Samples: 9

```
## Again we have to take the transpose of our data
pca <- prcomp(t(rna.data), scale=TRUE)

## Simple un polished plot of pc1 and pc2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```