Machine Learning 1

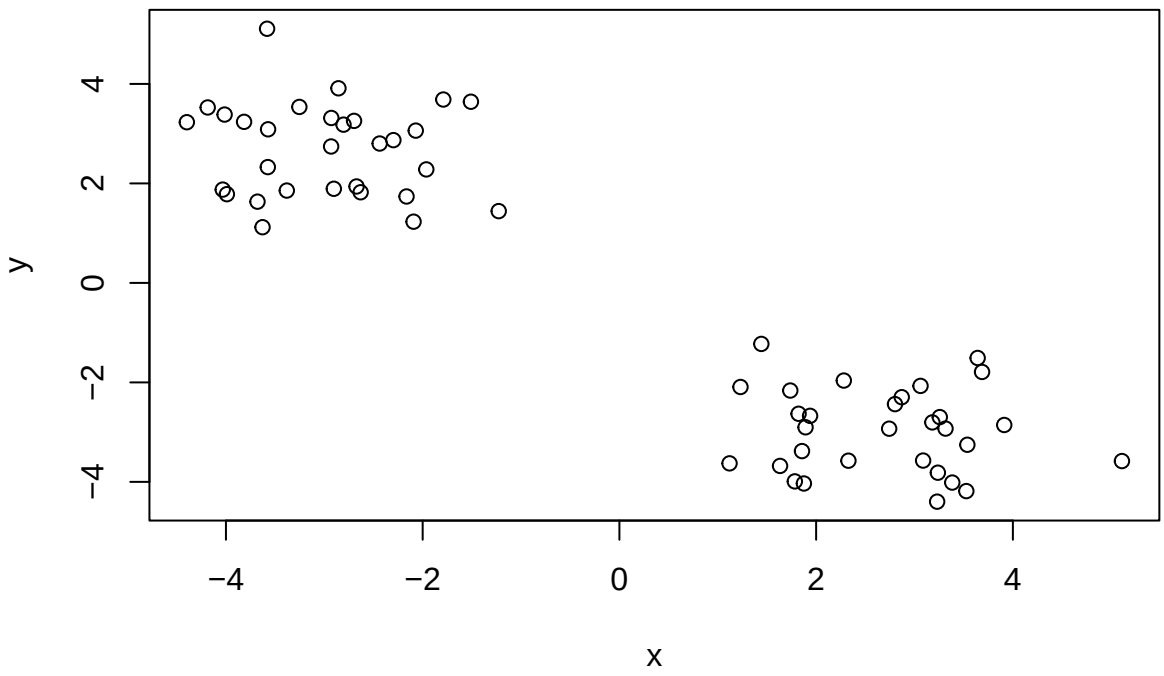
Tianru Zhang (PID: A15432834)

10/21/2021

First step is clustering methods #Kmeans clustering The function in the base R to do Kmeans clustering is called `Kmeans()`.

First, make up some data which nobody knows what the answer should be.

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



Q1. Can we do Kmeans() to cluster this data setting k=2 and nstart=20?

```
km<-kmeans(x,centers=2, nstart=20)

km
```

```
## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##           x         y
## 1  2.684145 -2.969310
## 2 -2.969310  2.684145
##
## 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 1 1
##
## Within cluster sum of squares by cluster:
## [1] 46.28487 46.28487
##  (between_SS / total_SS =  91.2 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Thus, we can do this clustering.

Q2. How many points are there in each cluster?

```
km$size
```

```
## [1] 30 30
```

Q3. What component of the result object details cluster assignment/membership?

```
km$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 1 1
```

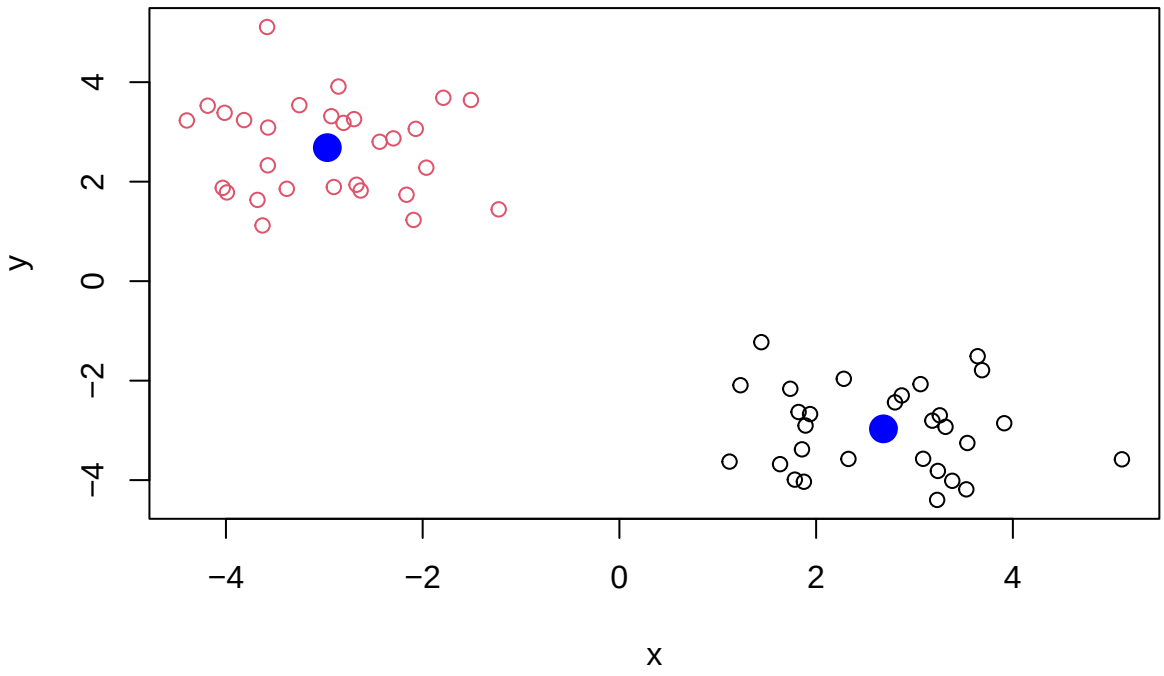
Q4. What component of the result object details the cluster centers?

```
km$centers
```

```
##           x         y
## 1  2.684145 -2.969310
## 2 -2.969310  2.684145
```

Q5 Plot x colored by the kmeans cluster assignment and add cluster centers as blue points

```
plot(x,col=km$cluster)
points(km$centers,col="blue",pch=16, cex=2)
```



#hclust

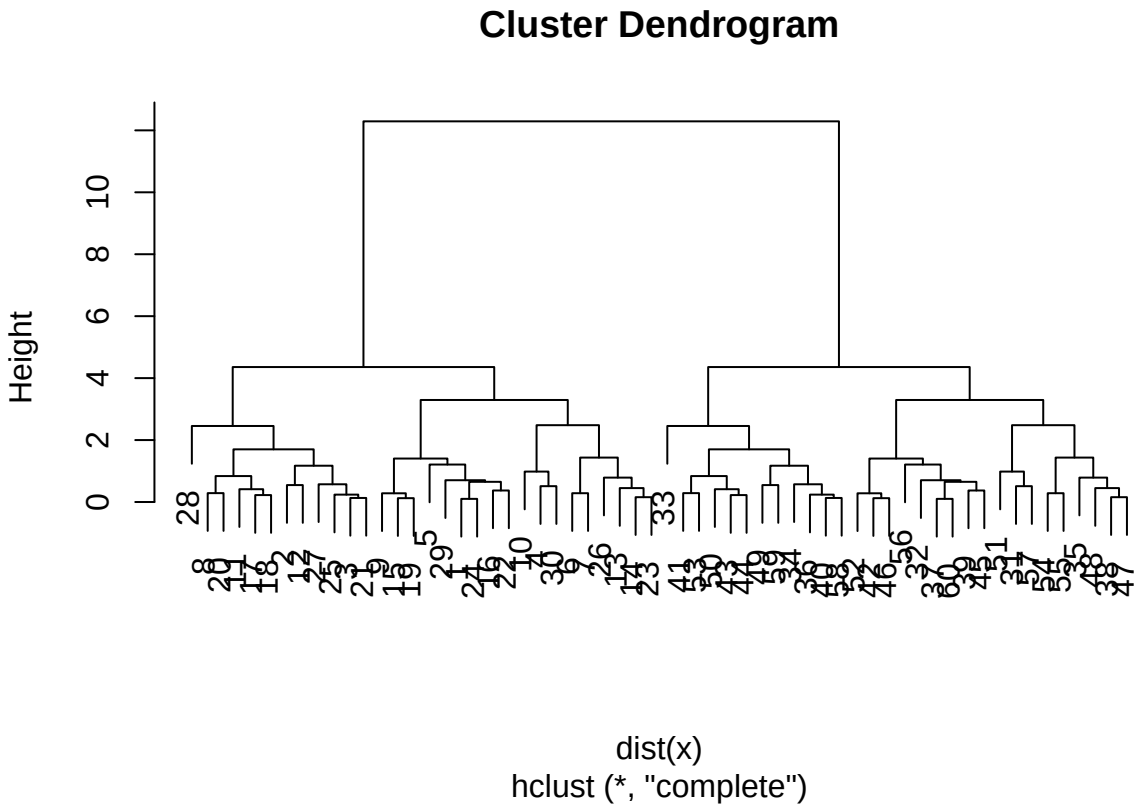
A big limitation with k-means is that we have to tell it the value of k(the number of clusters we want) Analyze with hclust, and demonstrate the use of dist(),hclust(),plot() and cutree() functions to do the clustering. Generate dendrograms and return cluster assignments to membership vectors.

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

```
##
## Call:
## hclust(d = dist(x))
##
## Cluster method   : complete
## Distance         : euclidean
## Number of objects: 60
```

There is a plot method for hclust result objects. Let's see it!!

```
plot(hc)
```



To get the clustering membership vector we have to do a bit more work. We have to cut the tree dendrogram where we consider making sense. Use the `cutree()`function

```
cutree(hc, h=6)
```

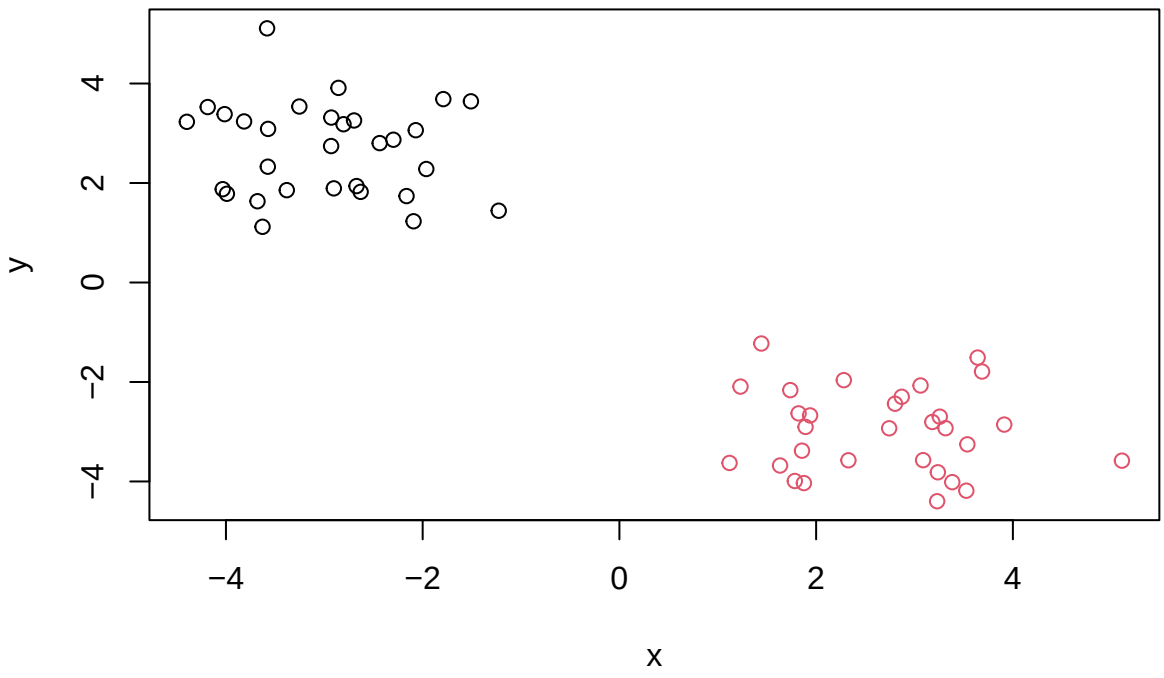
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

You can also call `cutree()` setting k=number of groups/clusters

```
grps<-cutree(hc, k=2)
```

Make the plot of groups

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




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

#Principal component data analysis(PCA)

##Getting 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?

```
dim(x)
```

```
## [1] 17  5
```

There are 17 rows and 5 columns. Whoops! There should only be four, because the first column is literally the names of rows.

```
head(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
```

The first way to fix the column name issue:

```
# 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
```

```
dim(x)
```

```
## [1] 17  4
```

The second way to fix:

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

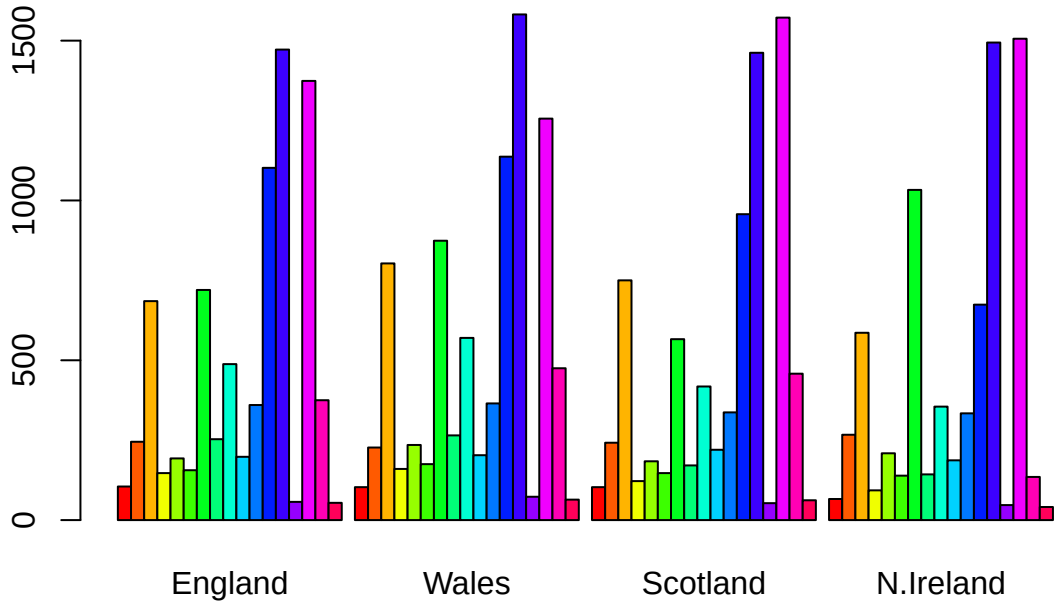
```
##                England Wales Scotland N.Ireland
## Cheese             105   103      103        66
## Carcass_meat       245   227      242       267
## Other_meat         685   803      750       586
## Fish               147   160      122        93
## Fats_and_oils      193   235      184 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

I prefer the second method, because with the first approach, if we run x <- x[,-1]), multiple times, it will lose column information and override the table.

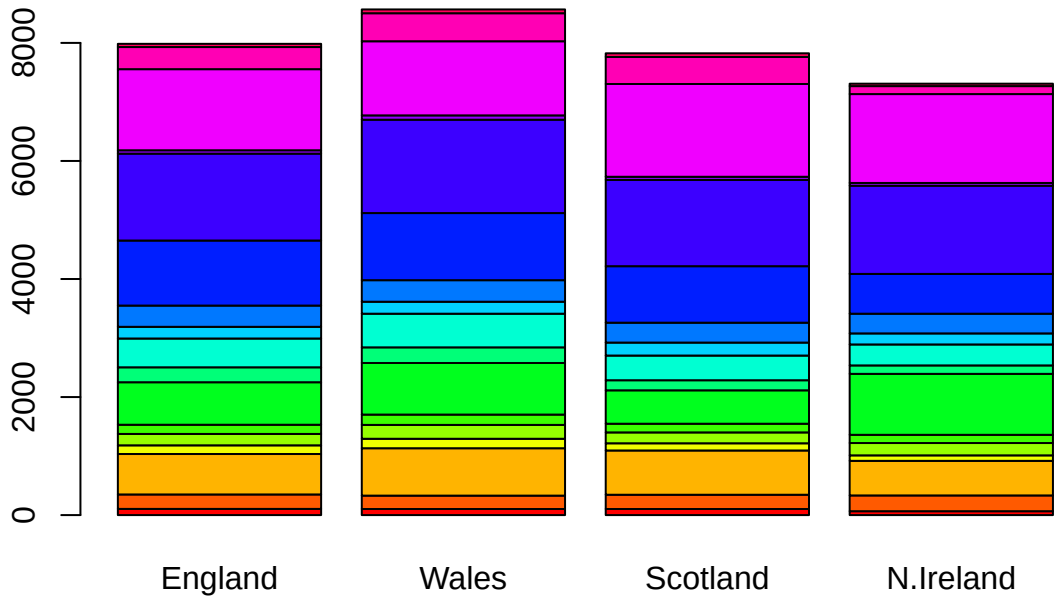
##Spotting major differneces and trends Now that the data looks good, we will analyze it with some conventional plots.

```
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?

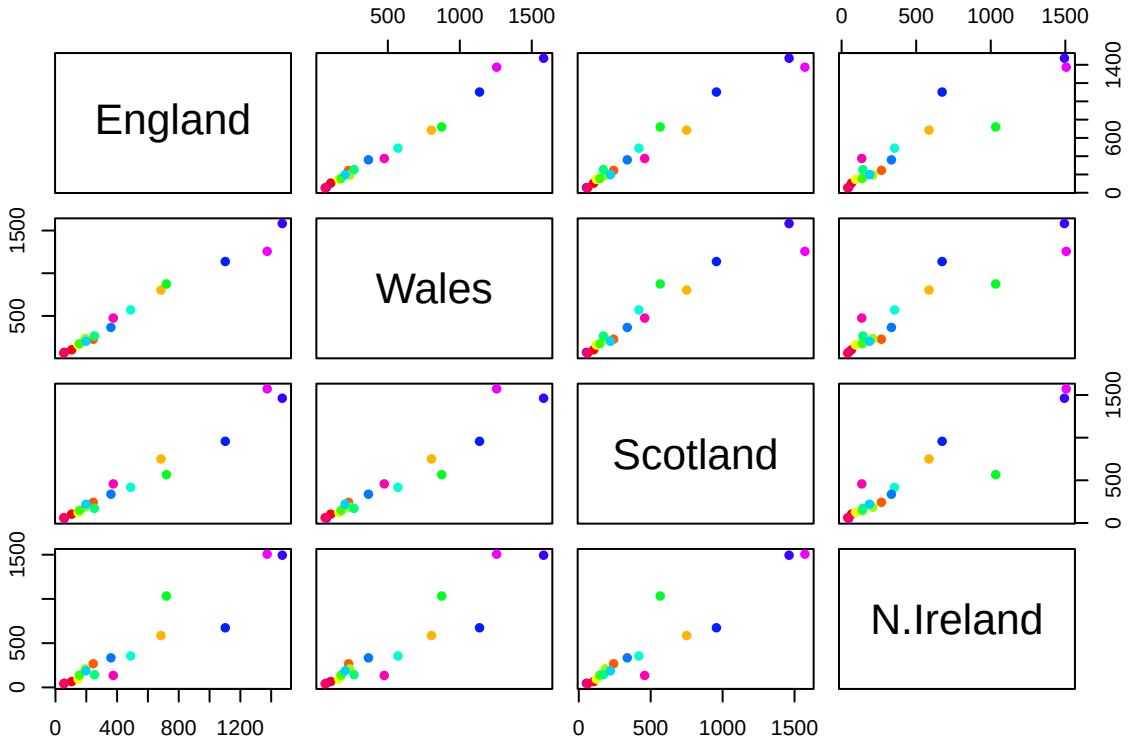
```
barplot(as.matrix(x), beside=FALSE, col=rainbow(nrow(x)))
barplot(as.matrix(x), col=rainbow(nrow(x)))
```



By setting the beside parameter to FALSE, or leaving the argument, will generate the new plot as above. The stacked plot is not helpful to look at.

> 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(17), pch=16)
```



This graph plots each country against each other. If a point lies on the diagonal, it means the two countries have the same amount of food consumption for that type of food.

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

The N. Ireland has one food consumption that significantly exceeds any other country in UK. But it's hrad to tell which food it is based on the plot alone.

#PCA to the rescue!

The main function in base R for PCA is `prcomp()`. This function transposes our data.

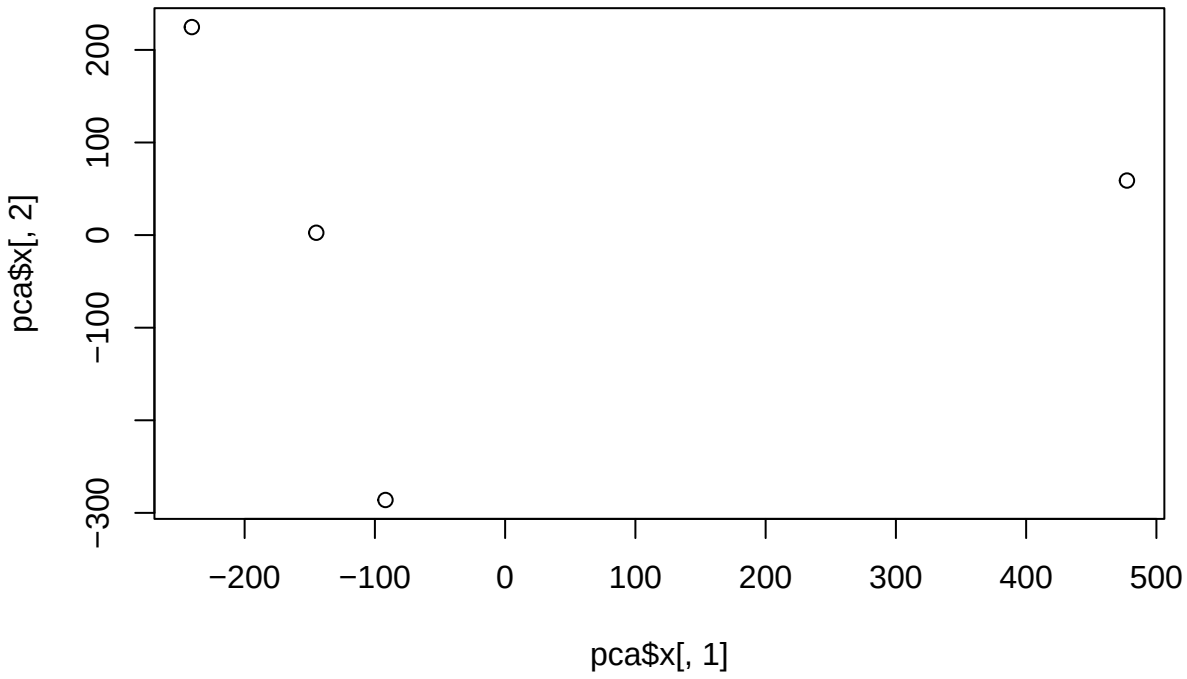
```
# Use the prcomp() PCA function
#t(x) takes the transpose of our data
pca <- prcomp( t(x) )
summary(pca)
```

```
## Importance of components:
##                             PC1      PC2      PC3        PC4
## Standard deviation     324.1502 212.7478 73.87622 4.189e-14
## Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
## Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

```
attributes(pca)
```

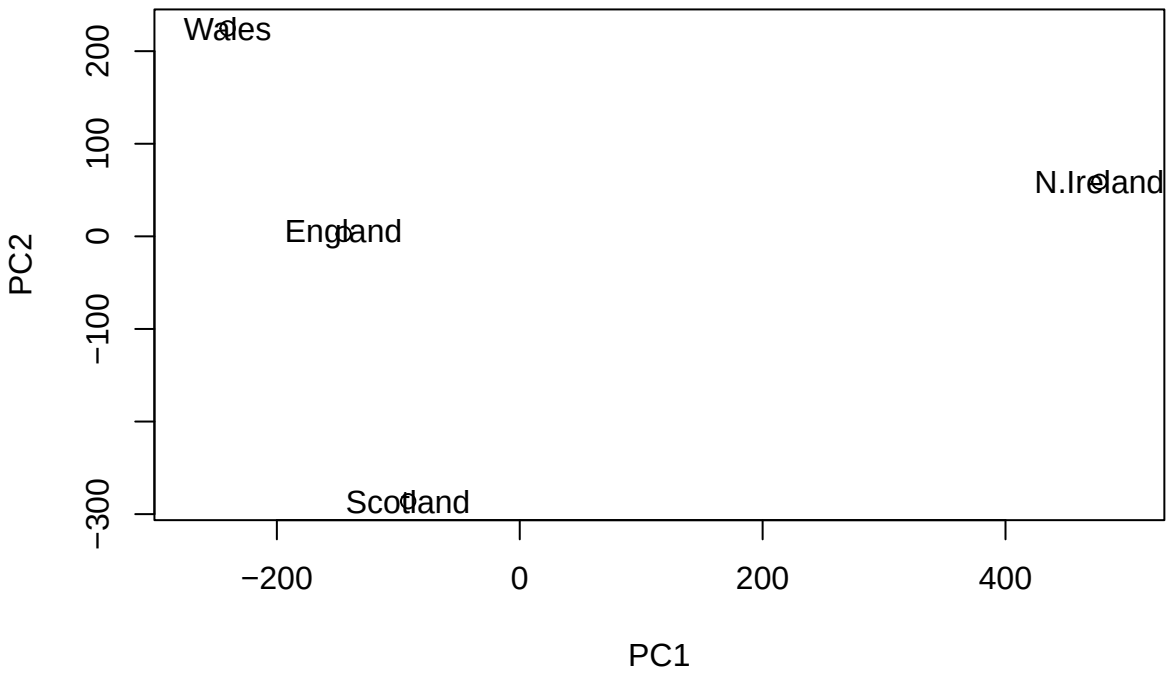
```
## $names
## [1] "sdev"     "rotation" "center"   "scale"    "x"
##
## $class
## [1] "prcomp"
```

```
plot(pca$x[,1],pca$x[,2])
```



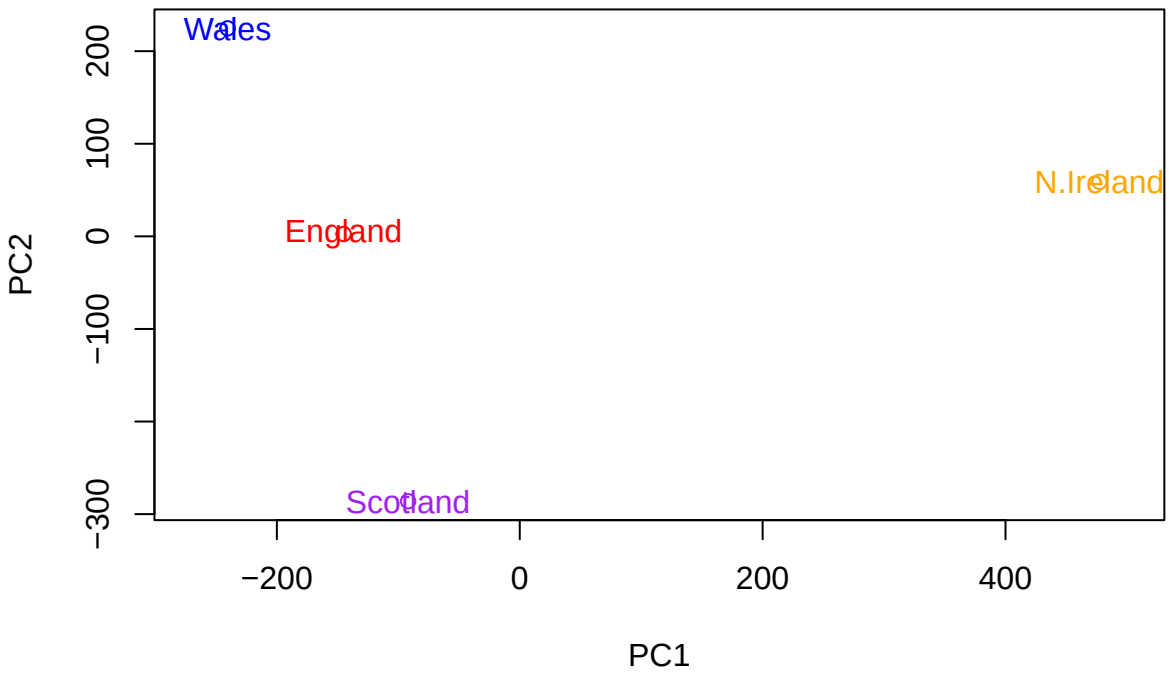
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

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



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.

```
color<-c("red","blue","purple","orange")
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500),col=color)
text(pca$x[,1], pca$x[,2], colnames(x),col=color)
```



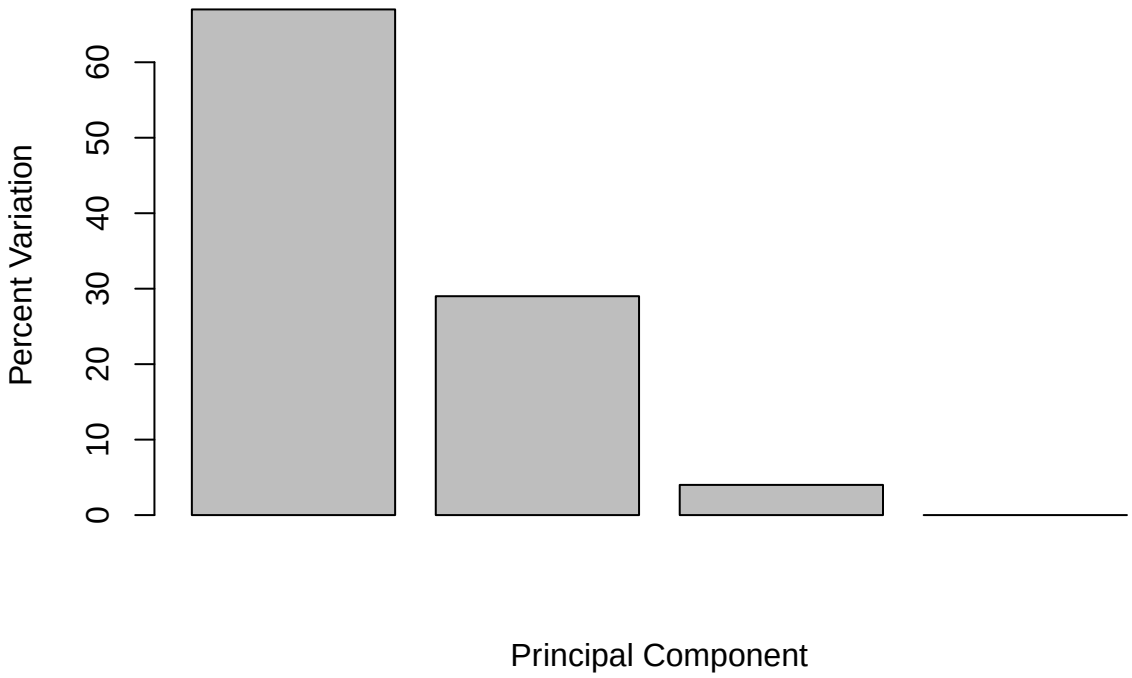

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

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

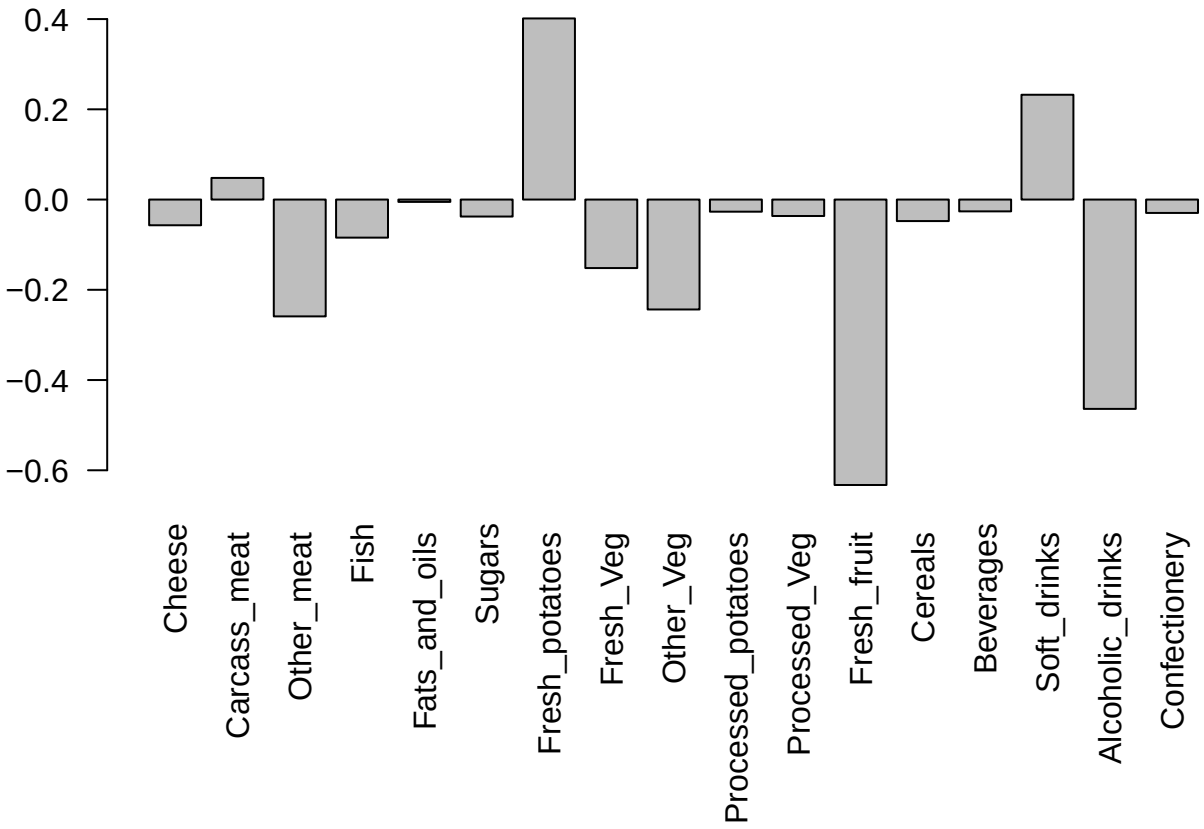
```
##                             PC1       PC2      PC3          PC4
## Standard deviation     324.15019 212.74780 73.87622 4.188568e-14
## Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
## Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

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



#Diving Deeper

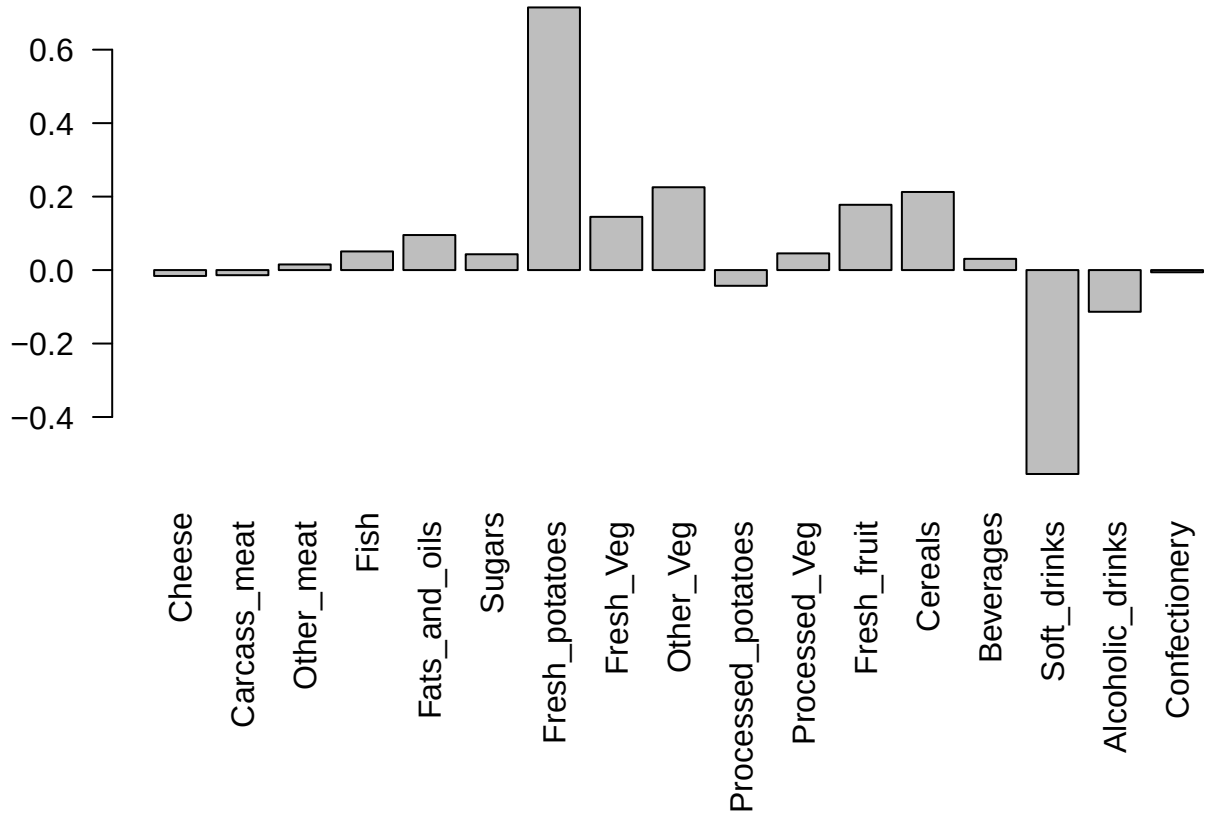
```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



This PC1 analysis shows difference in fresh fruit and alcoholic drinks

> Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about? Fresh potatoes and soft drinks prominantely shows difference in consumption.

```
## Lets focus on PC2 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```



#PCA of RNA-seq data

```
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?

```
dim(rna.data)
```

```
## [1] 100  10
```

There are 100 genes and 10 samples.