Machine Learning 1

Groot (PID: A15485151)

10/21/2021

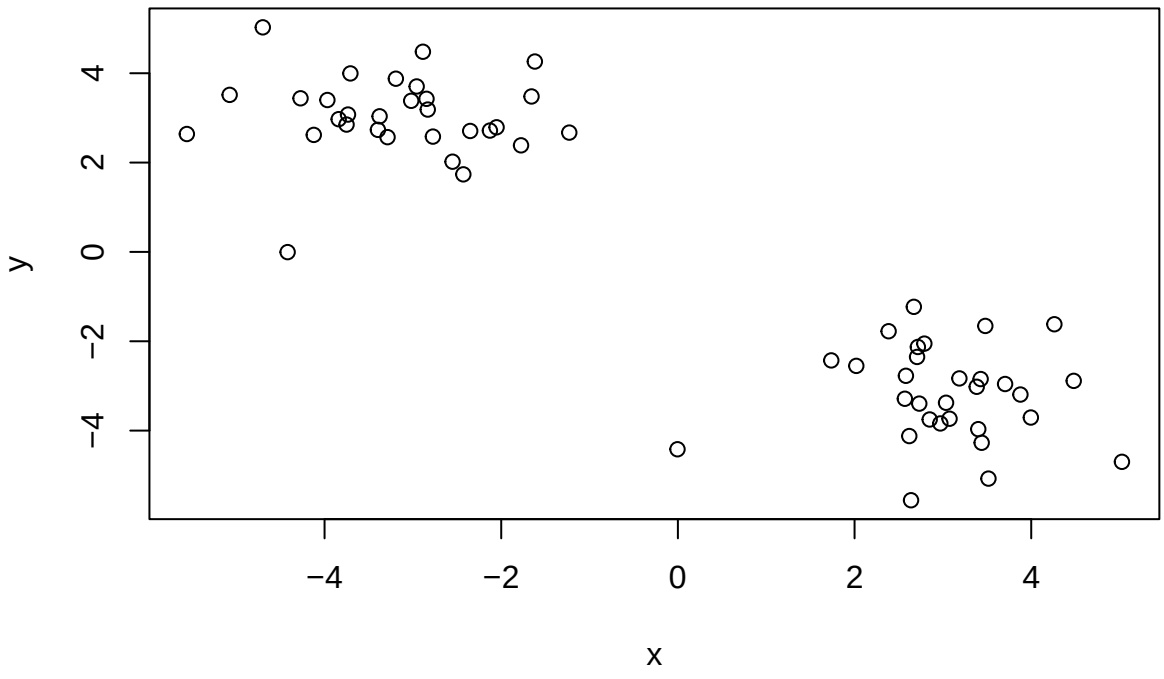
First up is clustering methods

Kmeans clustering

The function in base R to do Kmeans clustering is called 'kmeans()'.

First make up some data where we known what the answer should be:

```
# normal distribution around -3 and 3 (30 points each)
tmp <- c(rnorm(30,-3),rnorm(30,3))
x <- cbind(x=tmp, y=rev(tmp))
plot(x)
```



Q. Can we use 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 -3.183543  3.042565
## 2  3.042565 -3.183543
##
## 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 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 2
##
## Within cluster sum of squares by cluster:
## [1] 56.67982 56.67982
##  (between_SS / total_SS =  91.1 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Q. How many points are in each cluster?

```
km$size
```

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

Q. What 'component' of your result object details cluster assignment/membership?

```
km$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 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 2
```

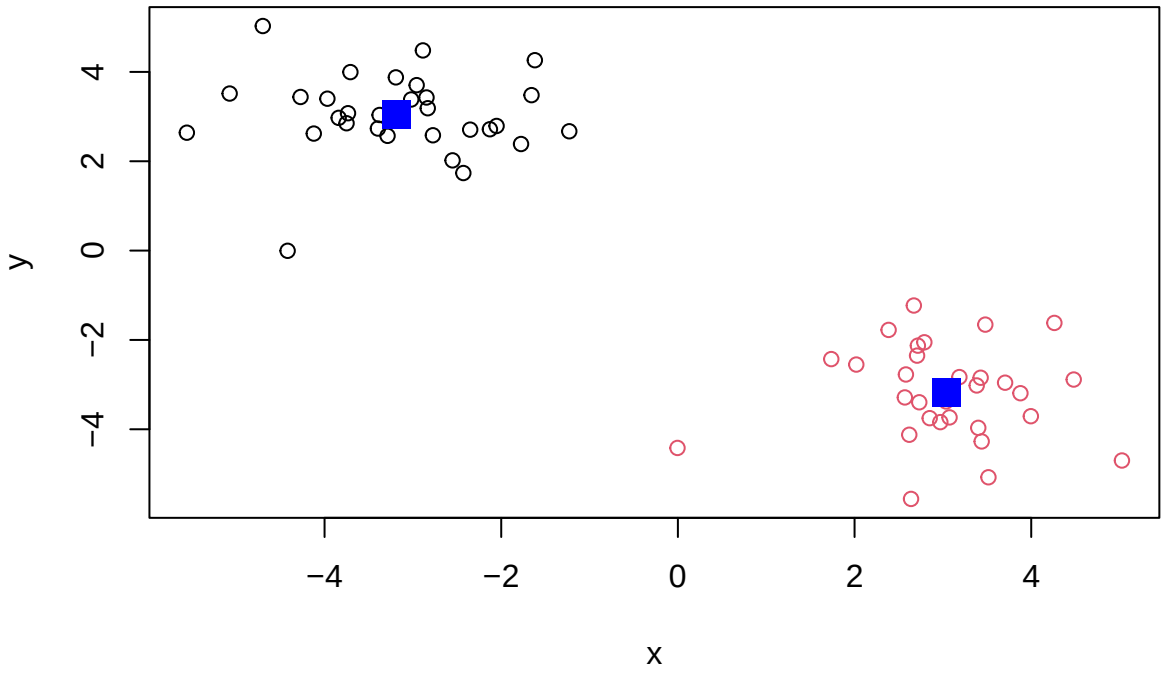
Q. What 'component' of your result object details cluster center?

```
km$centers
```

```
##           x         y
## 1 -3.183543  3.042565
## 2  3.042565 -3.183543
```

Q. 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=15, cex=2)
```




```
## integer(0)
```

Hierarchical clustering or hclust()

A big limitation with k-means is that we have to tell it K (the number of clusters we want).

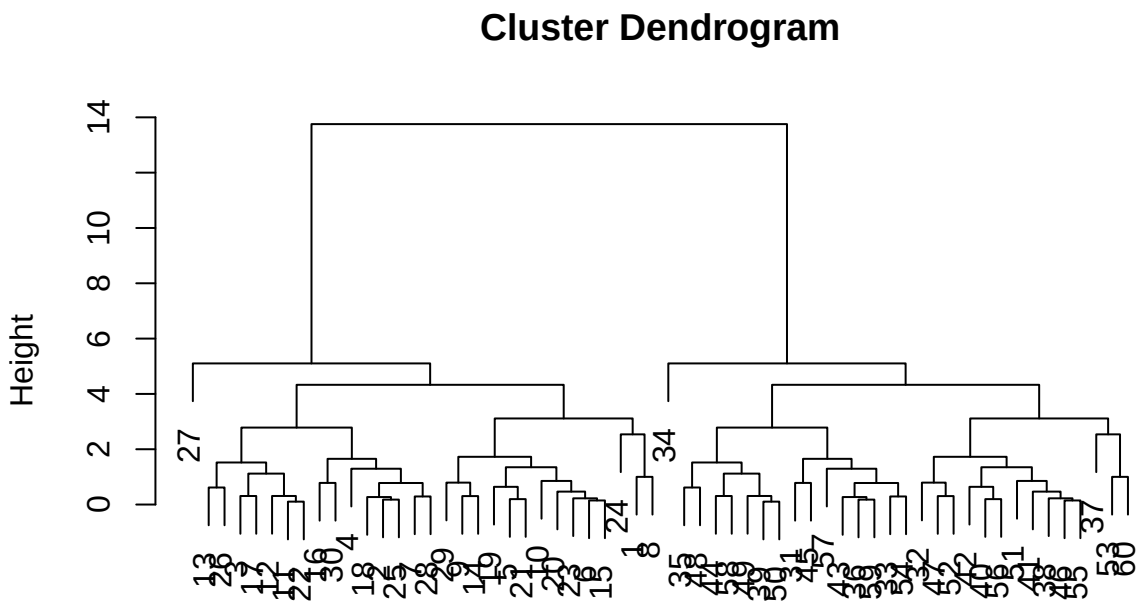
Use hclust instead to get around this.

```
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 hcust result objects.

```
plot(hc)
```



To get our cluster membership vector we have to do a little bit more work. We have to "cut" the tree (dendrogram) where we want. For this we use the 'cutree()' function.

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

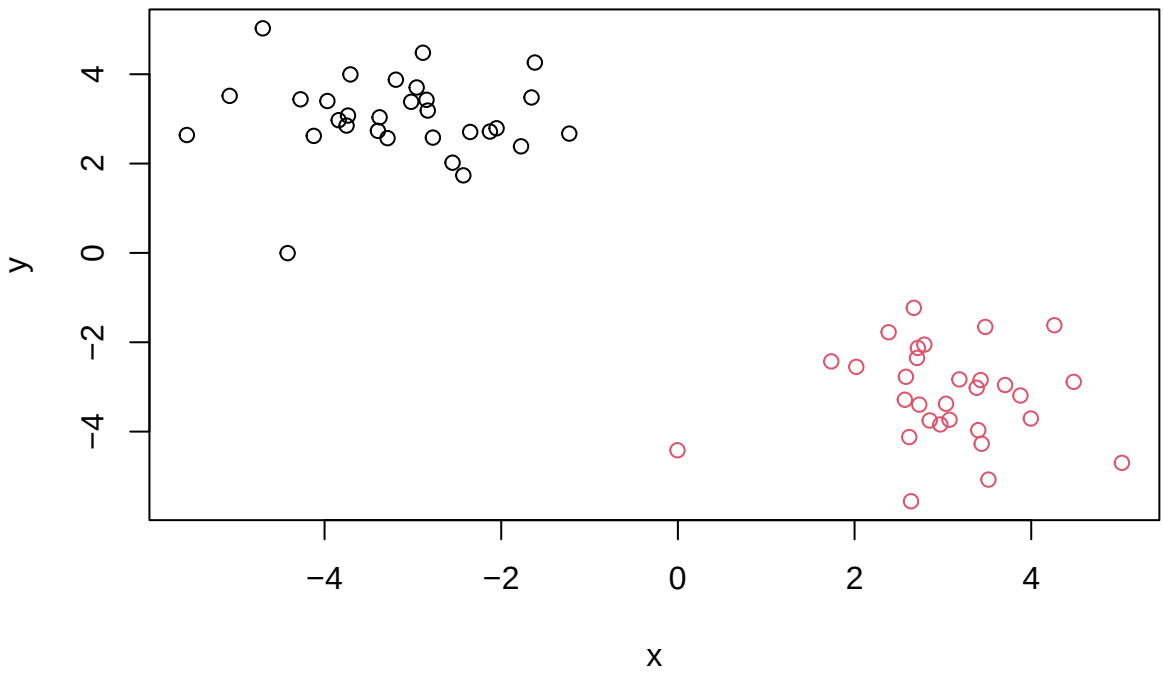
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

You can also call cutree() setting k=the number of groups/clusters you want

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

Make our results plot

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



Principal Component Analysis

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?

```
dim(x)
```

```
## [1] 60  2
```

Preview the first 6 rows

```
head(x)
```

```
##              x        y
## [1,] -5.074422 3.515709
## [2,] -3.019231 3.382798
## [3,] -2.428911 1.736303
## [4,] -2.886637 4.481024
## [5,] -3.287058 2.569276
## [6,] -3.840038 2.971180
```

Error in number of columns. Should be 17 x 4

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

```
## -5.07442167751919 -3.01923138805715 -2.42891109709952 -2.88663704894971
##          3.515709          3.382798          1.736303          4.481024
## -3.28705799582015 -3.84003782124459
##          2.569276          2.971180
```

We **lose** a country each time we run the code chunk... So instead

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names=1)
head(x)
```

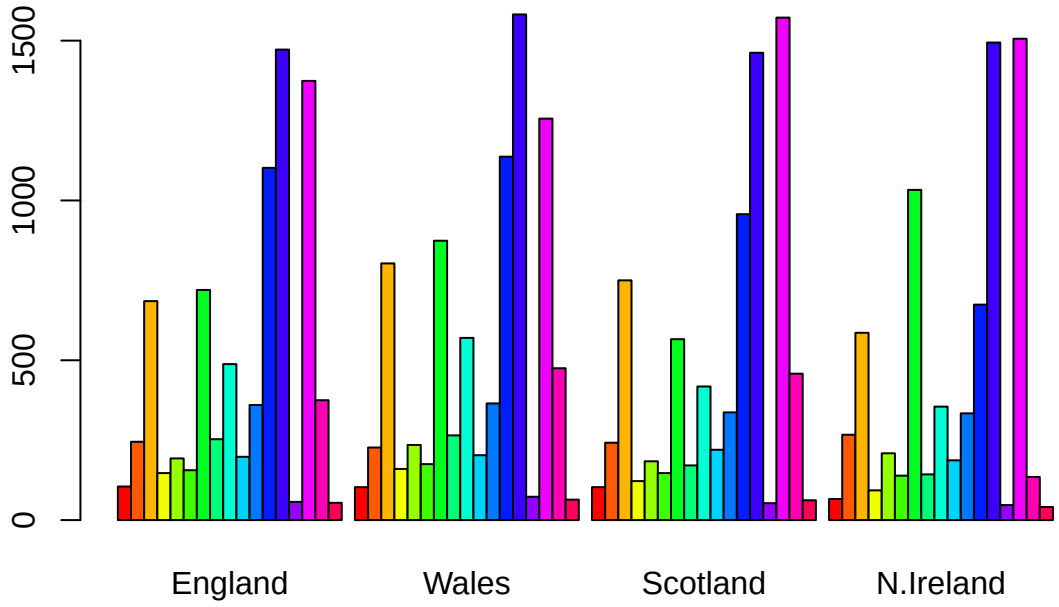
```
##                England Wales Scotland N.Ireland
## Cheese             105   103      103        66
## Carcass_meat       245   227      242       267
## Other_meat         685   803      750       586
## Fish               147   160      122        93
## Fats_and_oils      193   235      184       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 approach where you set the rownames ahead of time since it prevents the loss of data when the code chunk is run more than one time.

#Spotting major differences and trends

```
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?

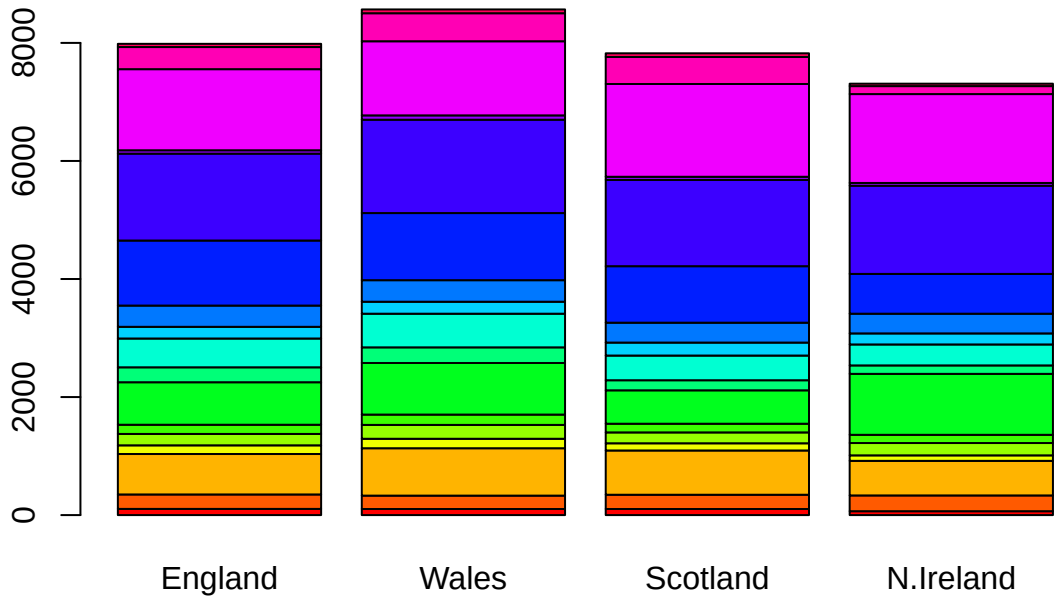
```
#set "besides = FALSE"
barplot(as.matrix(x), besides = FALSE,  col=rainbow(nrow(x)))
```

```
## Warning in plot.window(xlim, ylim, log = log, ...): "besides" is not a graphical
## parameter
```

```
## Warning in axis(if (horiz) 2 else 1, at = at.l, labels = names.arg, lty =
## axis.lty, : "besides" is not a graphical parameter
```

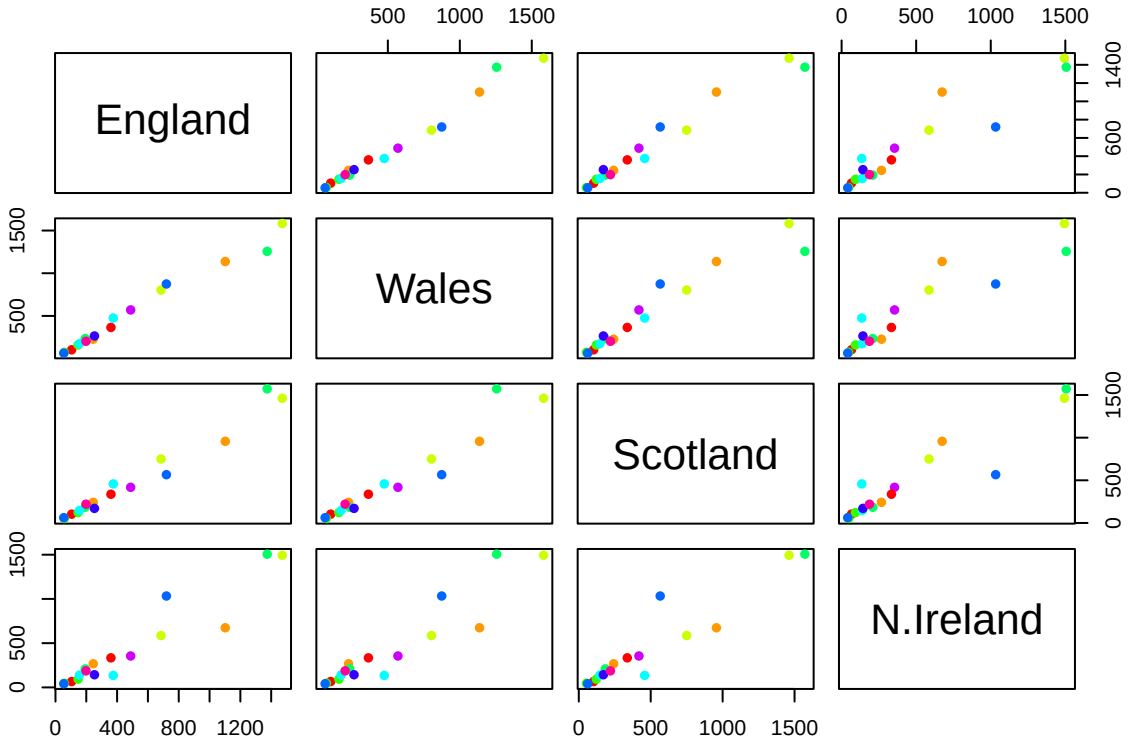
```
## Warning in title(main = main, sub = sub, xlab = xlab, ylab = ylab, ...):
## "besides" is not a graphical parameter
```

```
## Warning in axis(if (horiz) 1 else 2, cex.axis = cex.axis, ...): "besides" is not
## a graphical parameter
```



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)
```



If a point lies on the diagonal for the given plot, it means that there is a strong correlation between the two countries for that same food type. This means that the more the points lie close to the diagonal line, the more similar the two countries in question are in terms of what the people of each country eat.

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

The main differences between N. Ireland and the other countries of the UK is the blue plot that deviates from the diagonal in all three plots against the country of N. Ireland.

A lot of tedious work #PCA to the rescue!

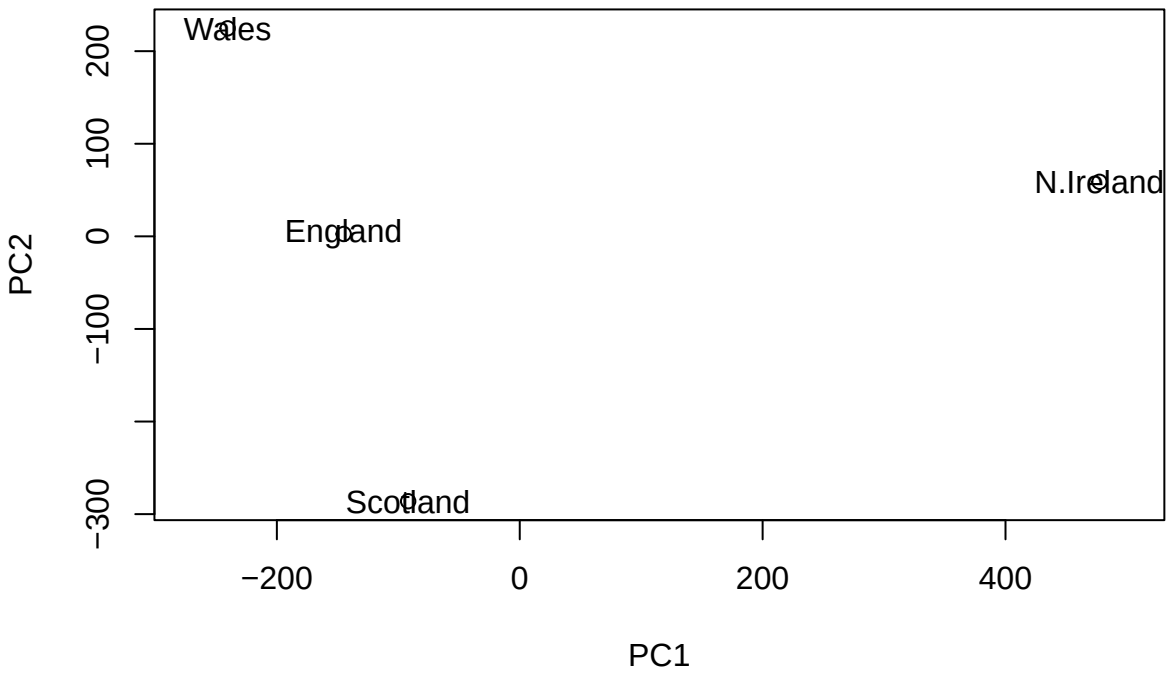
The main function in base R for PCA is 'prcomp()' This requires us to transpose (switch x and y) the data w/ "(t(x))"

```
# Use the prcomp() PCA function
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
```

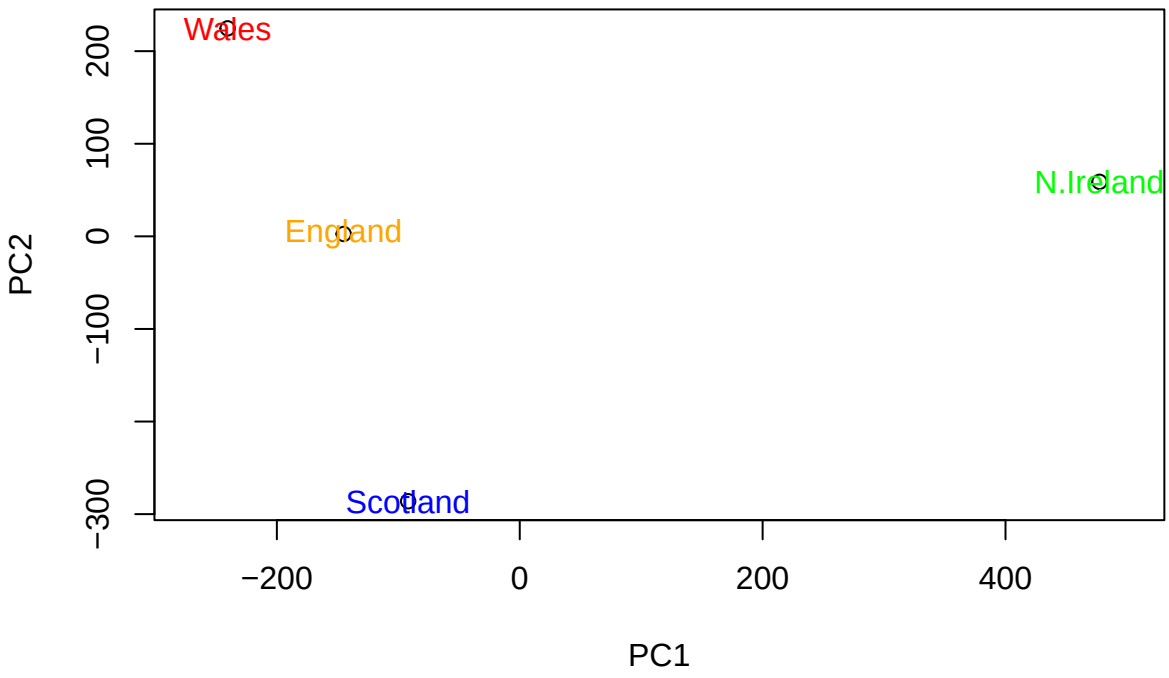
> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
# 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.

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

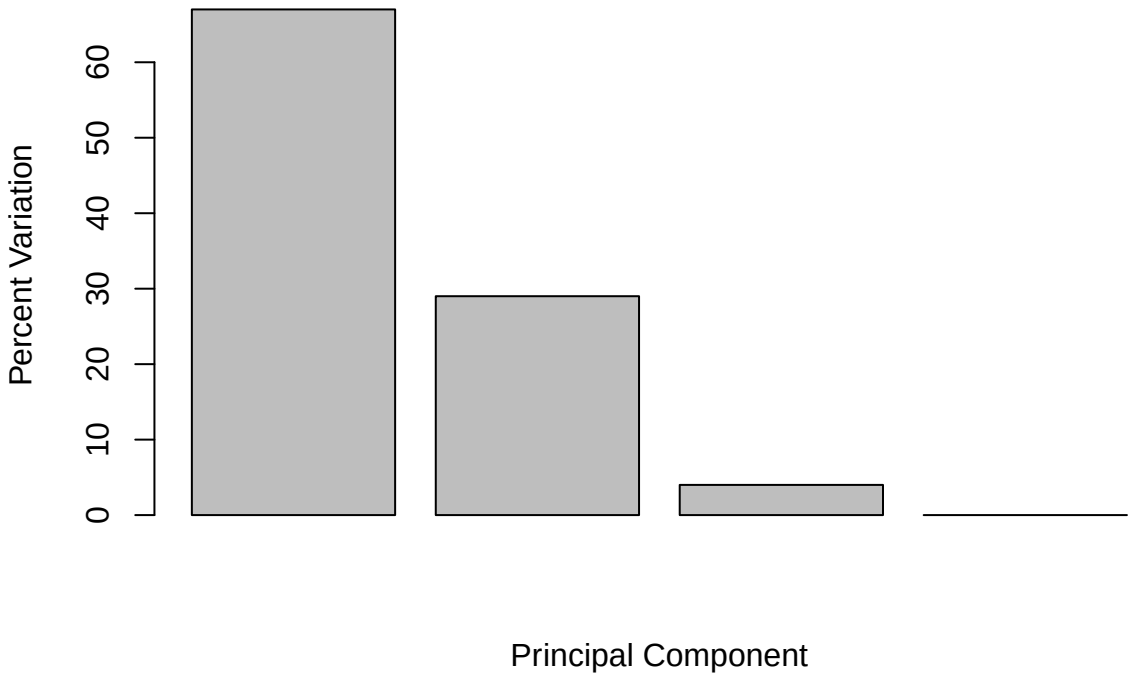


variation level for pca

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

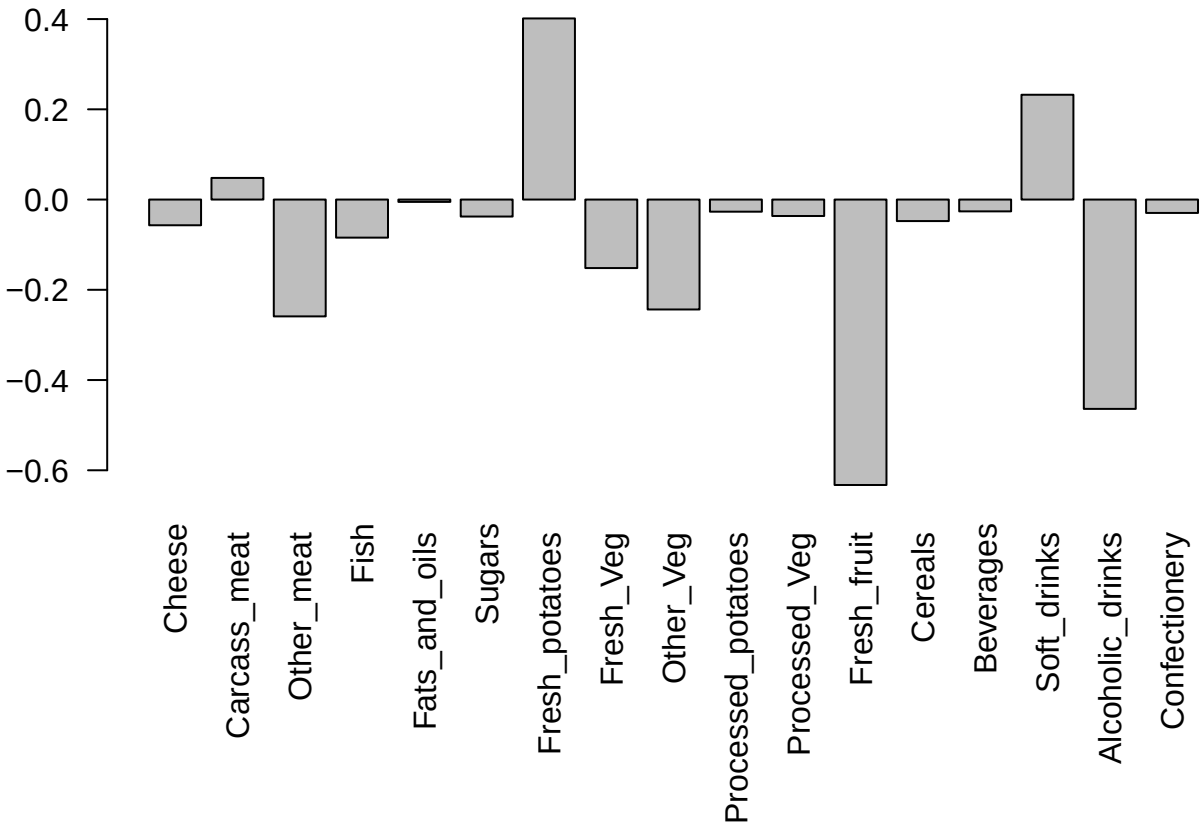
```
## [1] 67 29  4  0
```

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



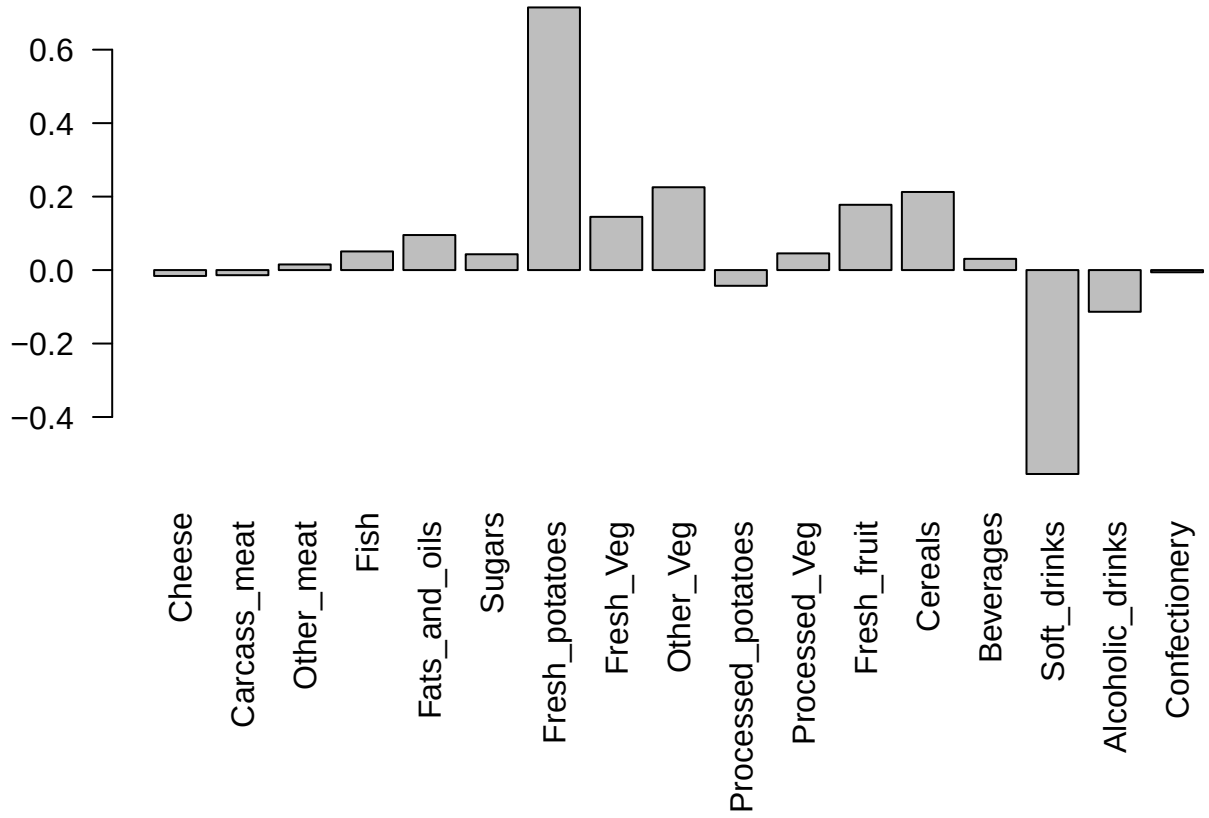
#variable loading

```
## 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 )
```



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

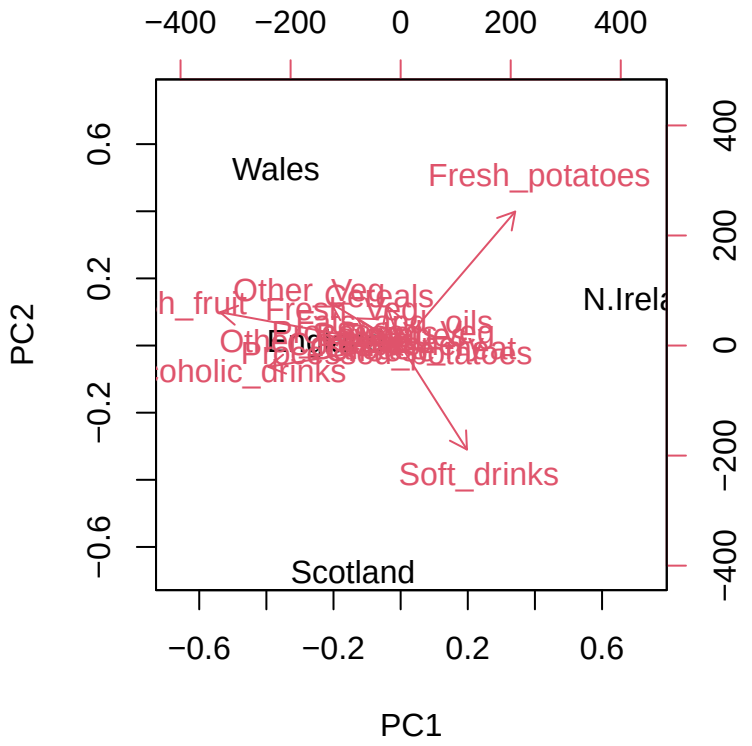
```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```



The two food groups that are featured predominantly is fresh potatoes and soft drinks. PC2 tells us about most of the variation that is not covered by PC1.

#Biplots

```
## The inbuilt biplot() can be useful for small datasets
biplot(pca)
```



#PCA of RNA-seq data load 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
```

100 genes and 10 samples for each gene!