Class 7: Machine Learning 1

Brianna Smith

In this class we will explore clustering and dimentionality reduction methods.

K-means

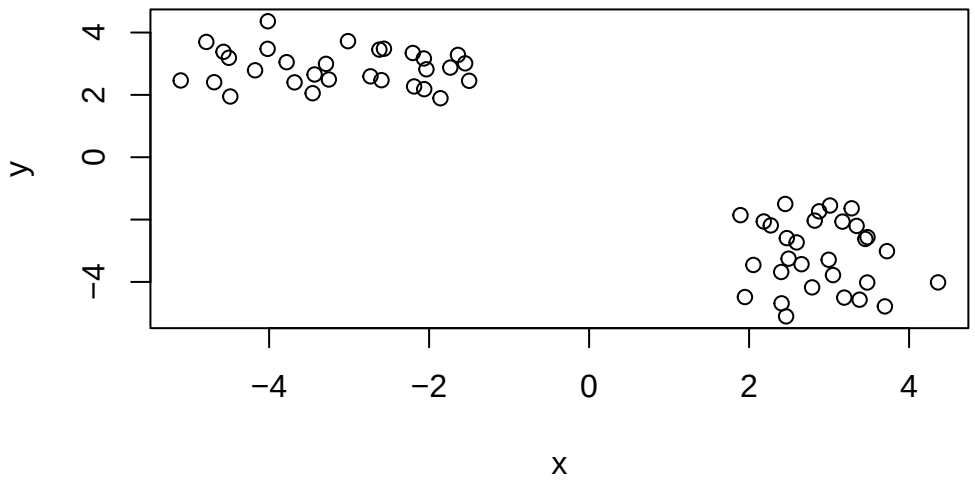
Make up some input data where we know what the answer should be.

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

```
            x        y
[1,] -2.621652 3.452759
[2,] -3.253355 2.493472
[3,] -4.485373 1.947802
[4,] -5.104990 2.462535
[5,] -4.015770 4.362386
[6,] -4.687104 2.406280
```

Quick plot of x to see the two groups at -3,+3 and +3,-3

```
plot(x)
```



Use the `kmeans()` function setting k to 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  2.879388 -3.120082
2 -3.120082  2.879388

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] 45.83673 45.83673
 (between_SS / total_SS =  92.2 %)

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?
- cluster center?

```
km$cluster
```

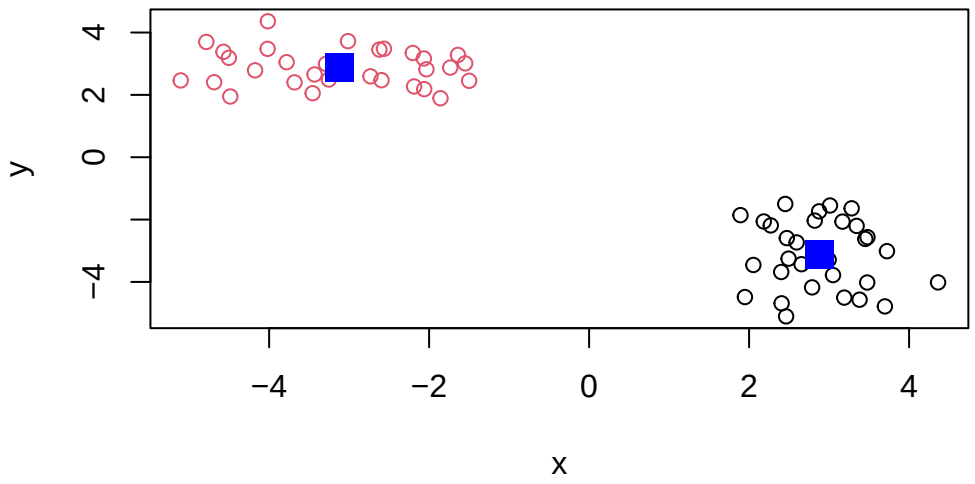
```
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
km$centers
```

```
          x         y
1  2.879388 -3.120082
2 -3.120082  2.879388
```

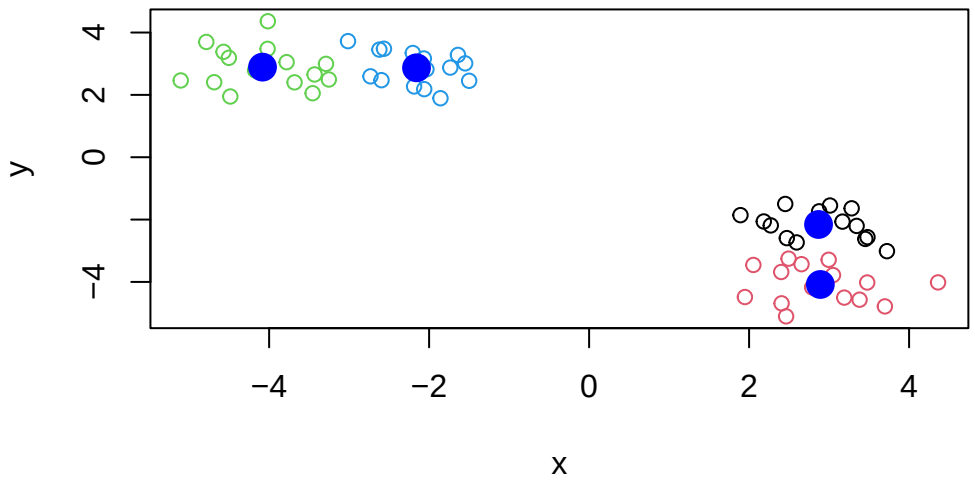
Q. Plot x colored by the kmeans cluster assignment and add cluster center as blue points?

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



Play with kmeans and ask for different number of clusters

```
km <- kmeans(x, centers=4, nstart=20)
plot(x, col=km$cluster)
points(km$centers, col="blue", pch=16, cex=2)
```



Heirarchial Clistering

This is another very useful and widely employed clustering method which has the advantage over k-means in that it can help reveal the something of the true grouping in your data.

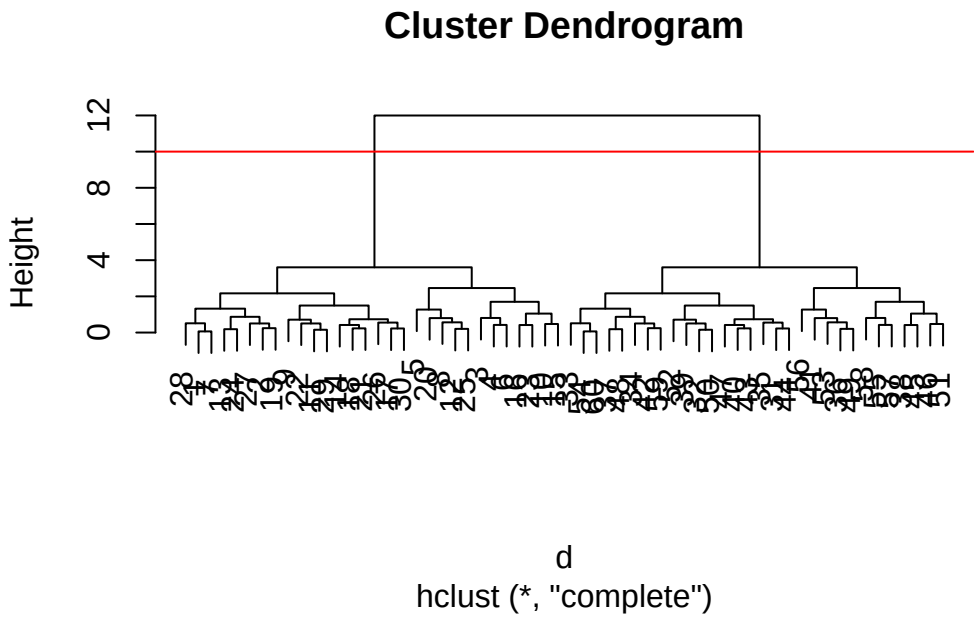
The `hclust()` function wants a distance martix as input. We can get this from the `dist()` function.

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

```
Call:
hclust(d = d)

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

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

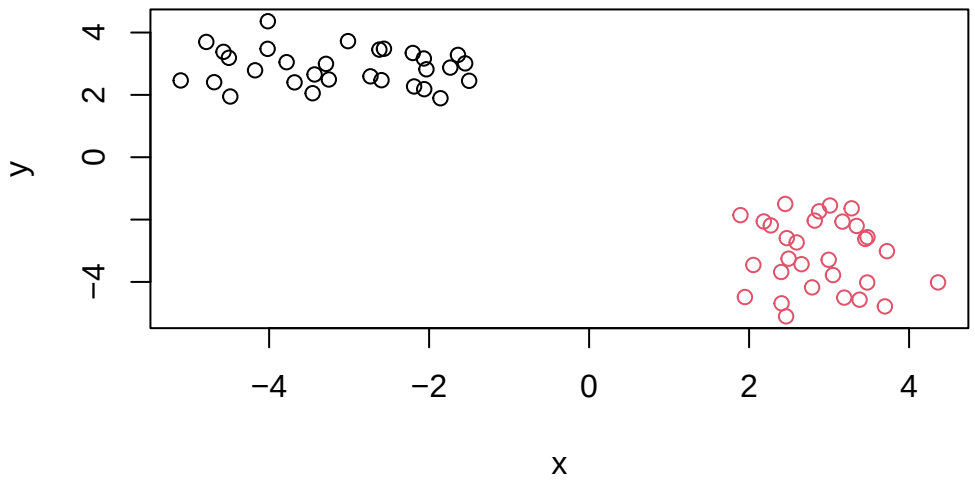


To get my cluster membership vector I need to "cut" my tree to yeild sun-trees or branches with all the members of a given cluster residing on the same cut branch. The function to do this is called `cutree()`

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

```
 [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 2 2
```

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



It is often helpful to use the `k=` argument to cutree rather than the `h=` height of cutting with `cutree()`. This will cut the tree to yeild the number of clusters you want.

```
cutree(hc, k=4)
```

```
 [1] 1 1 2 2 2 2 1 2 1 2 1 2 1 1 2 2 1 1 1 2 1 1 2 1 2 1 1 1 1 1 3 3 3 3 3 4 3 4
[39] 3 3 4 3 3 3 4 4 3 3 4 3 4 3 4 3 4 4 4 4 3 3
```

#Principal Component Analysis (PCA)

The base R function for PCA is called `prcomp()`

```
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
```

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
```

```
nrow(x)
```

```
[1] 17
```

```
ncol(x)
```

```
[1] 5
```

Preview the first 6 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
```

```
# 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
```

```
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
```

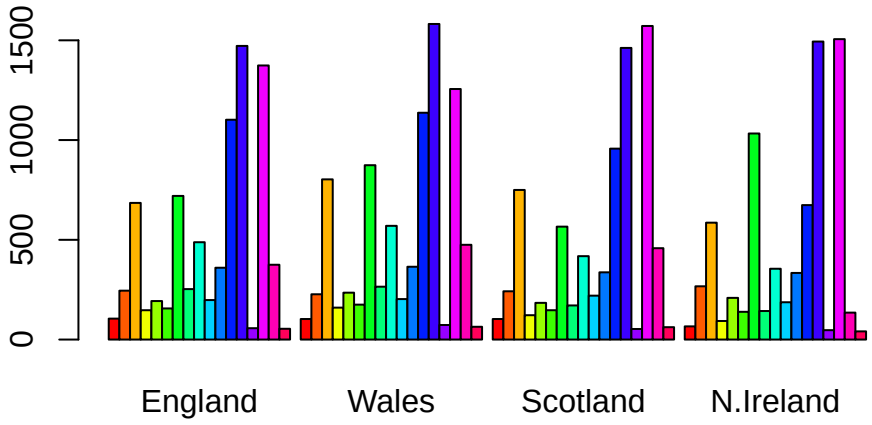
```
dim(x)
```

```
[1] 17  4
```

> 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?

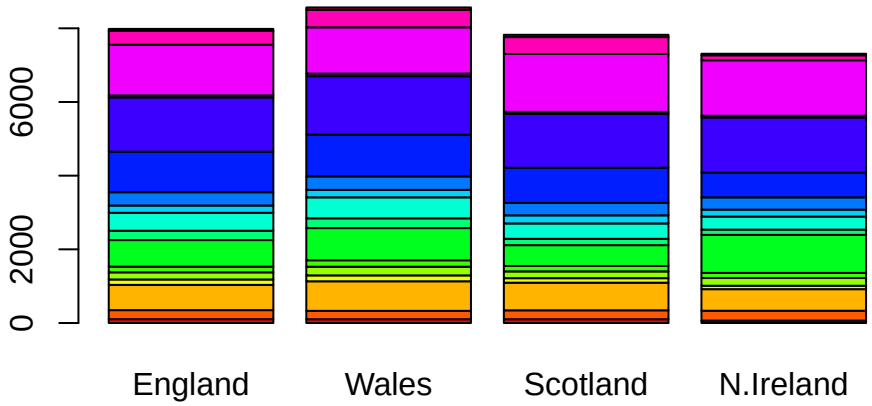
The second approach is more robust. With the first method, every time you run the code it deletes a column. This does not fix the problem entirely, the way that the second method does. No matter how many times you run the code, the second method will give the proper number of columns. The second method is prefered.

```
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?

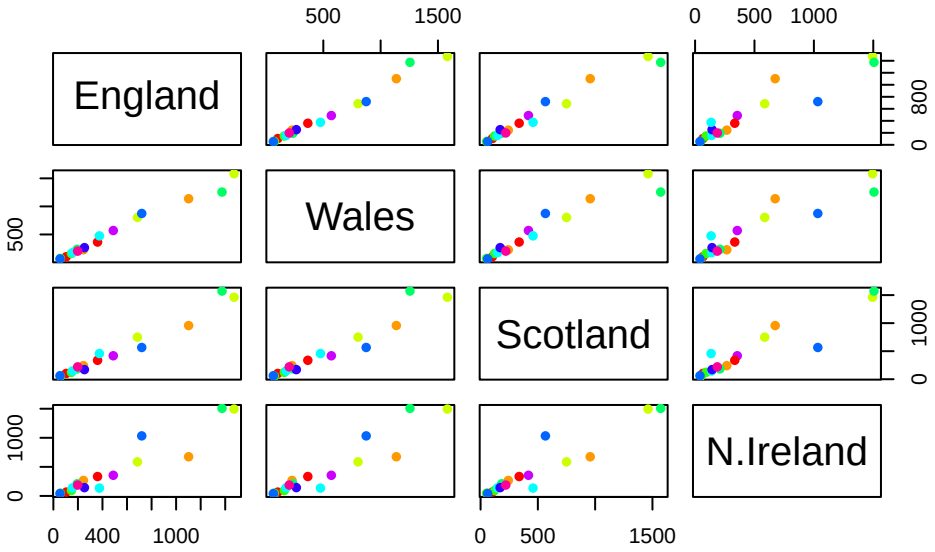
```
barplot(as.matrix(x), beside=F, col=rainbow(nrow(x)))
```



You would change the T to an F in the barplot code

> 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)
```



The plot compares all the pairs in separate graphs. If the plot shows a diagonal pattern, or if a point in the graph lies diagonal, this indicates that the two countries being compared have a similar numerical value for that category.

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

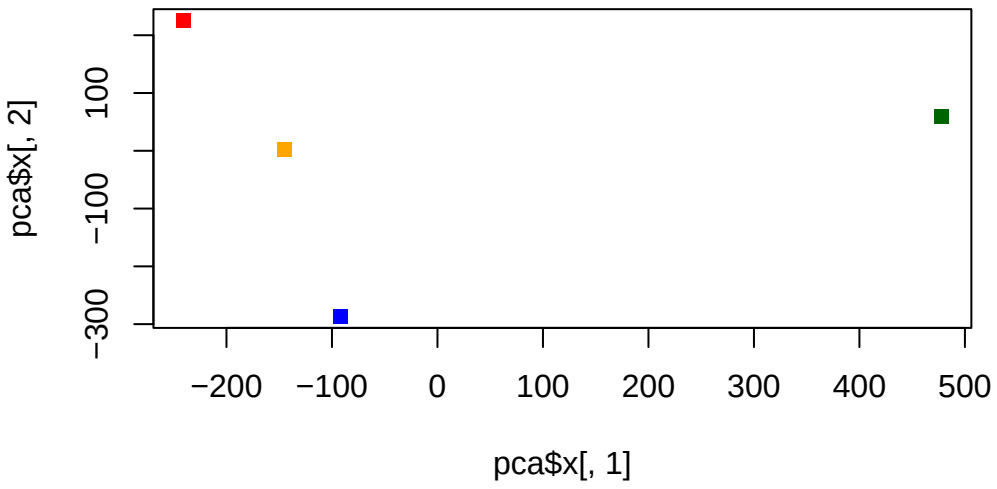
The main difference that N. Ireland has when compared to the other countries is the dark blue and the orange points on the graph because they are further from the diagonal line in the data.

```
# 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
```

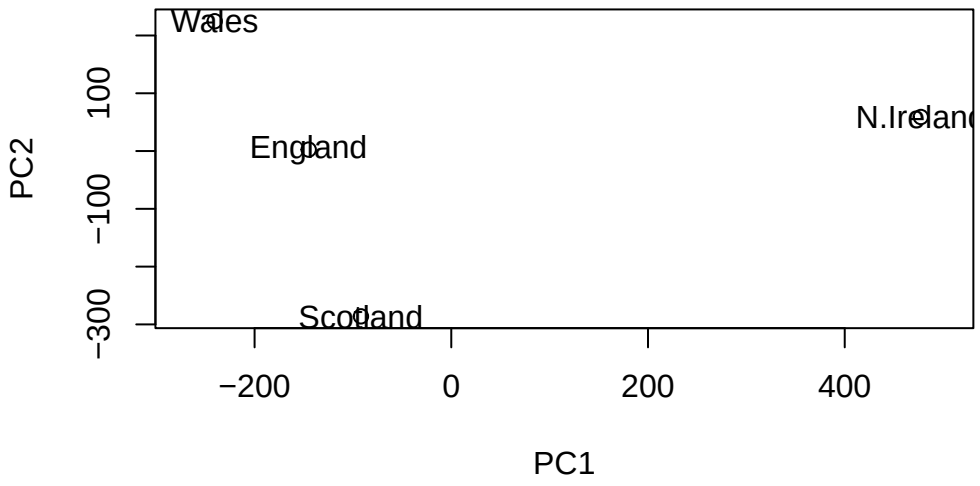
A “PCA plot” (a.k.a. “Score plot”, PC1vsPC2 plot, etc.)

```
plot(pca$x[,1], pca$x[,2], col=c("orange", "red", "blue", "darkgreen"), pch=15)
```



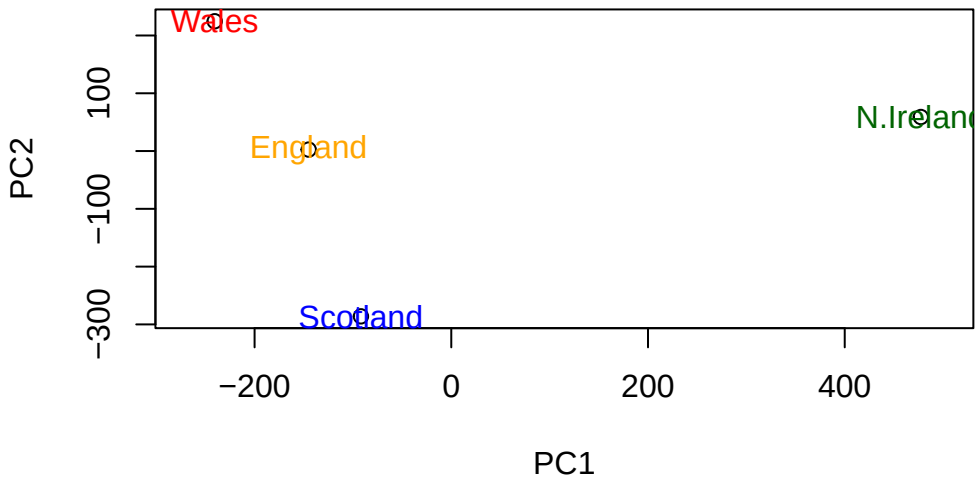
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
# 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 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), col=c("orange", "red", "blue", "darkgreen"), pch=1
```



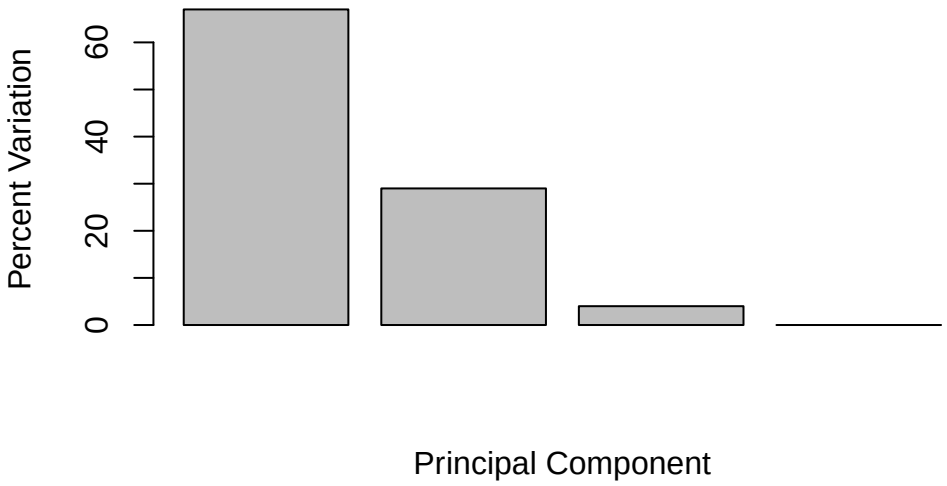

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

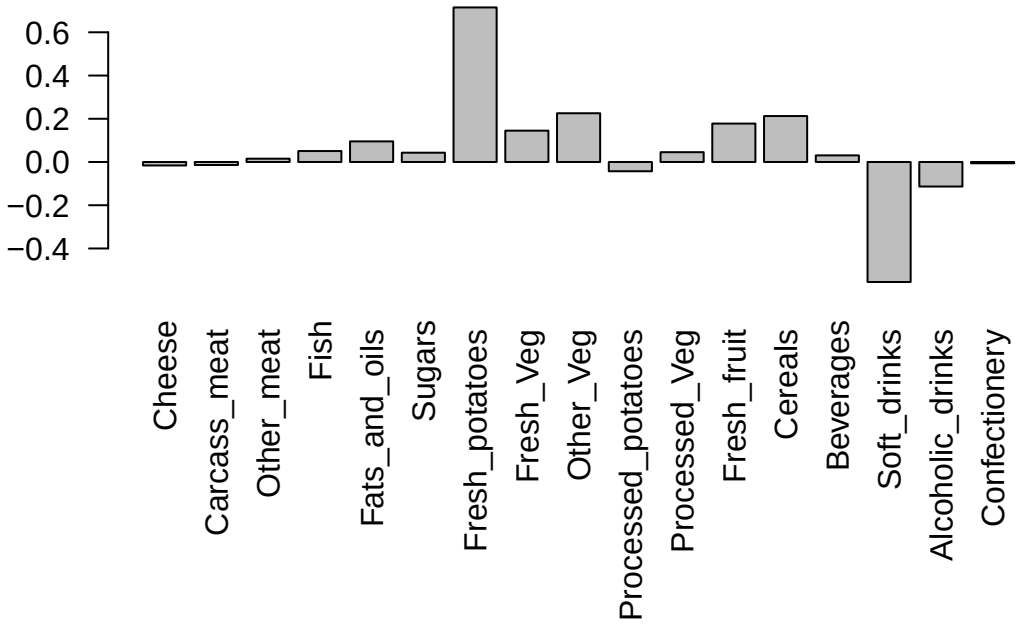
```
## or the second row here...
z <- summary(pca)
z$importance
```

```
                             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")
```




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


0.6
0.4
0.2
0.0
−0.2
−0.4
Cheese
Carcass_meat
Other_meat
Fish
Fats_and_oils
Sugars
Fresh_potatoes
Fresh_Veg
Other_Veg
Processed_potatoes
Processed_Veg
Fresh_fruit
Cereals
Beverages
Soft_drinks
Alcoholic_drinks
Confectionery