Class 7: Machine Learning

Aparajita Pranjal

4/26/23

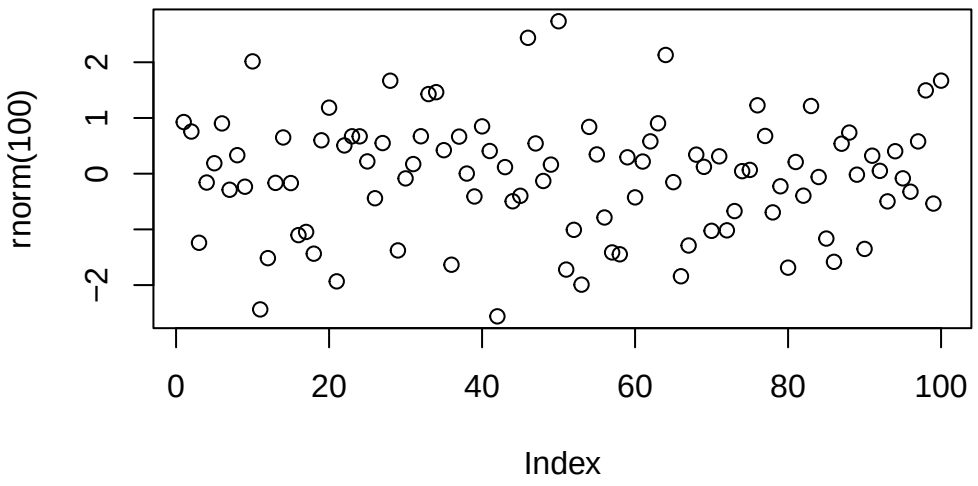
Example of K-means clustering

First step is to make up some data with a known structure, so we know what the answer should be.

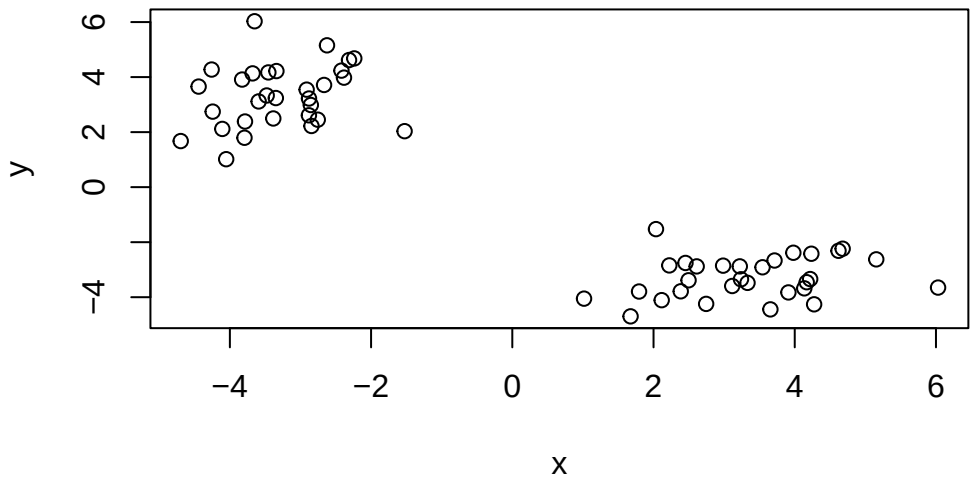
```
rnorm(10)
```

```
 [1] -0.04409999  0.67330625 -1.54672339  2.52448535  0.13165324  1.69679529
 [7] -1.22999086 -0.19408450 -0.08546699 -1.46381027
```

```
plot(rnorm(100))
```




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



Now we have some structured data in `x`. Let's see if k-means is able to identify the two groups.

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

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

Cluster means:
          x         y
1  3.325356 -3.280938
2 -3.280938  3.325356

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] 53.61536 53.61536
 (between_SS / total_SS =  92.4 %)

Available components:
```

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

Let's explore `k` :

```
k$size
```

```
[1] 30 30
```

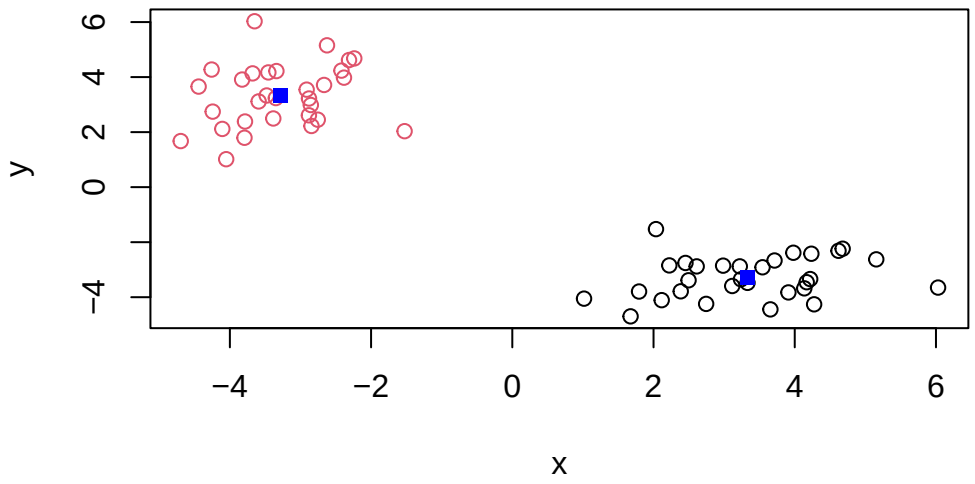
```
k$centers
```

```
          x         y
1  3.325356 -3.280938
2 -3.280938  3.325356
```

```
k$cluster
```

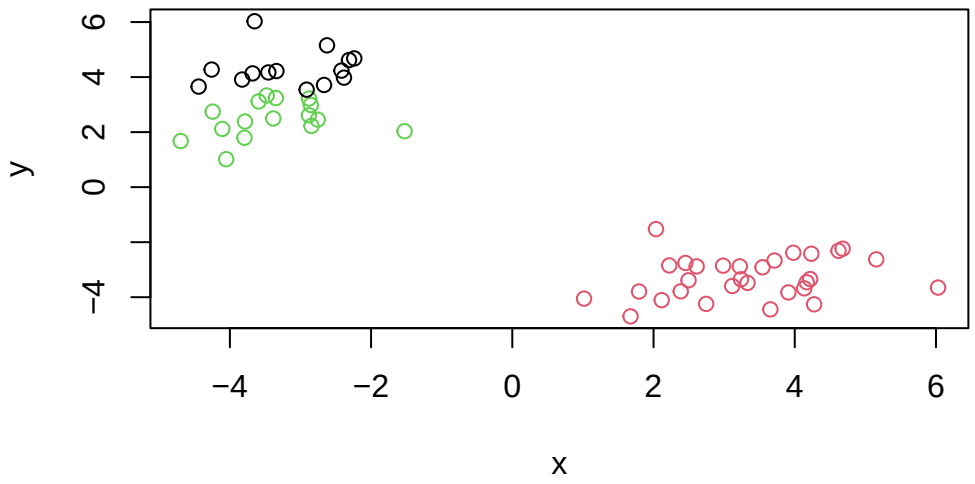
```
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
plot(x, col = k$cluster)
points(k$centers, col = 'blue', pch = 15)
```



Example with wrong number of clusters for k-means:

```
k_3 <- kmeans(x, centers = 3, nstart = 20)
plot(x, col = k_3$cluster)
```



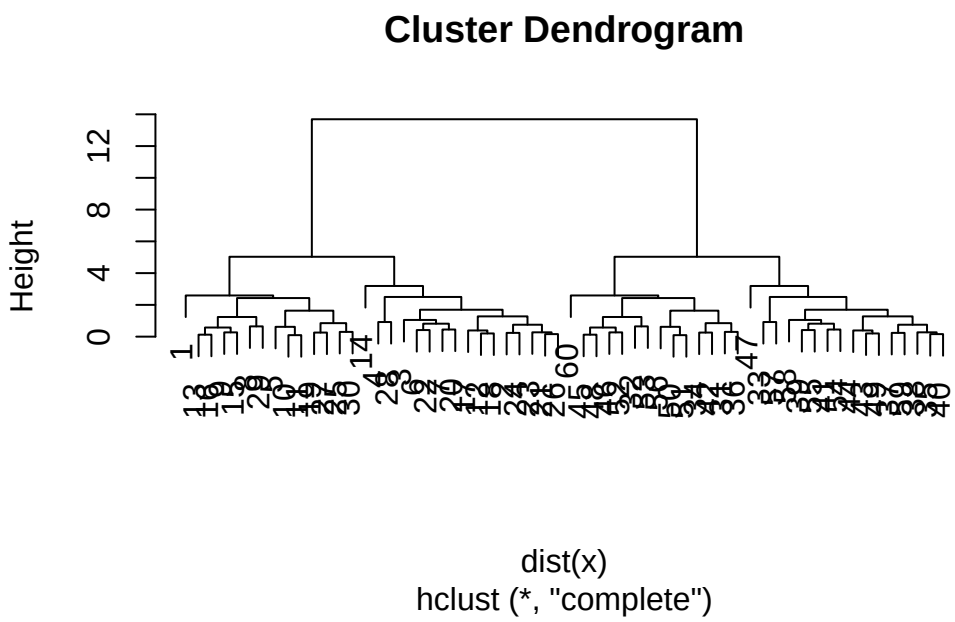
Example of Hierarchical Clustering

Let's use the same data stored in `x` and the `'hclust()'` function:

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

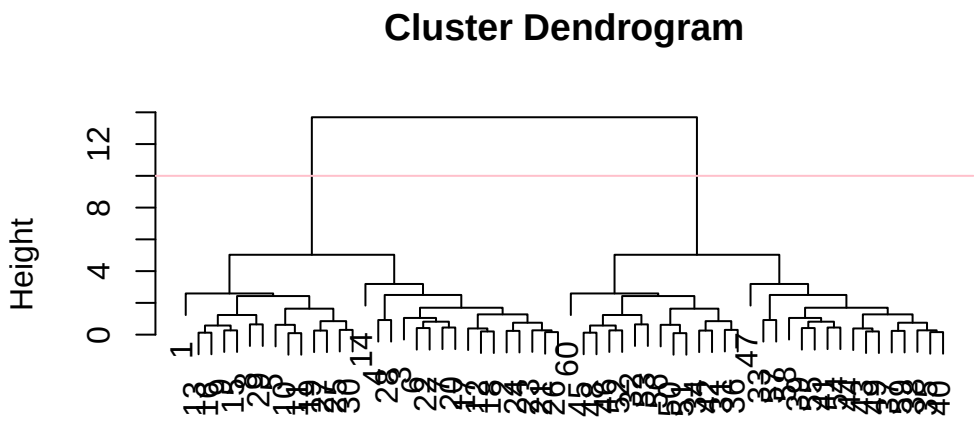
```
Call:
hclust(d = dist(x))

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

```
plot(clustering)
```



Let's add a horizontal line

```
plot(clustering)
abline(h = 10, col = 'pink')
```

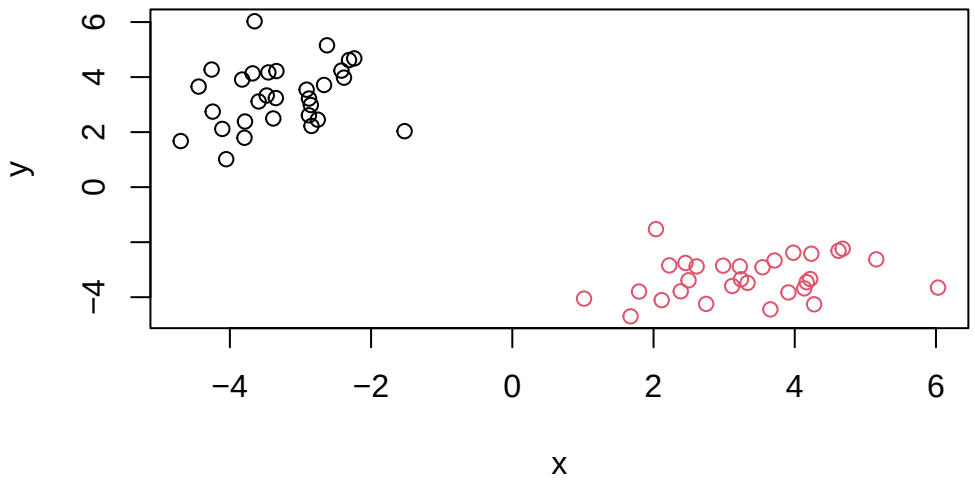


To get our results (i.e., membership vector) we need to "cut" our tree. The function for doing that is `'cutree()'`.

```
subgroups <- cutree(clustering, h = 10)
subgroups
```

```
 [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 = subgroups)
```



You can also "cut" your tree with the number of clusters (k):

```
cutree(clustering, k = 2)
```

```
 [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
```

Principal Component Analysis

The aim is to reduce dimensionality (or surfaces) while only losing a small amount of information. The first axis shows the higher variability and the second axis will show less and so on.

PCA of UK Food

Data import

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names = 1)
# header = F will remove name of columns
```

```
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
```

Q1. How many rows and columns are in your new data frame named `x`? What R functions could you use to answer this questions? **17 rows and 4 columns**

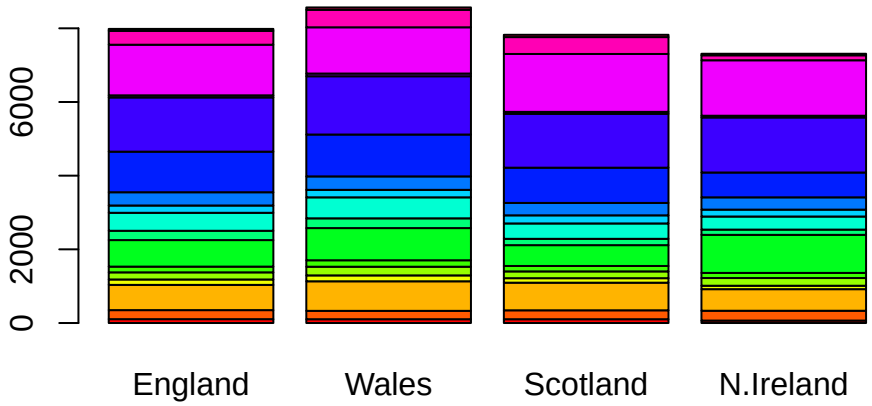
Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances? **The function `row.names` is more robust than negative indexing as that will remove the first column every time it is run so more data will be removed. Whereas `row.names` only removes the column we choose.**

```
dim(x)
```

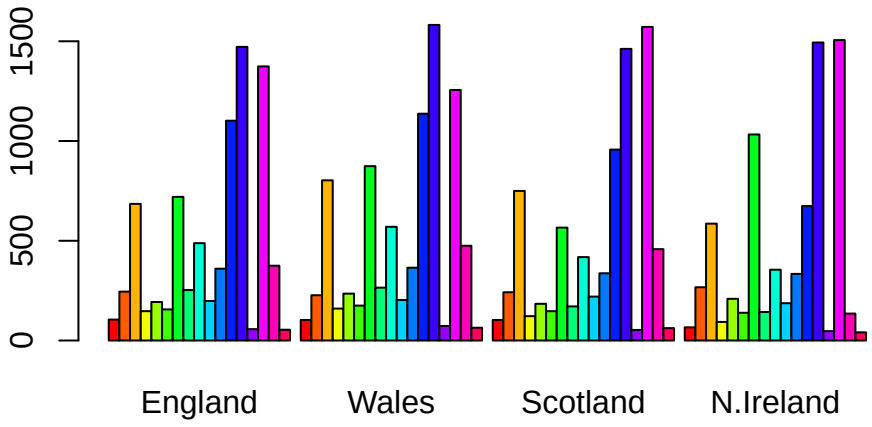
```
[1] 17  4
```

Now we can generate some basic visualizations:

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



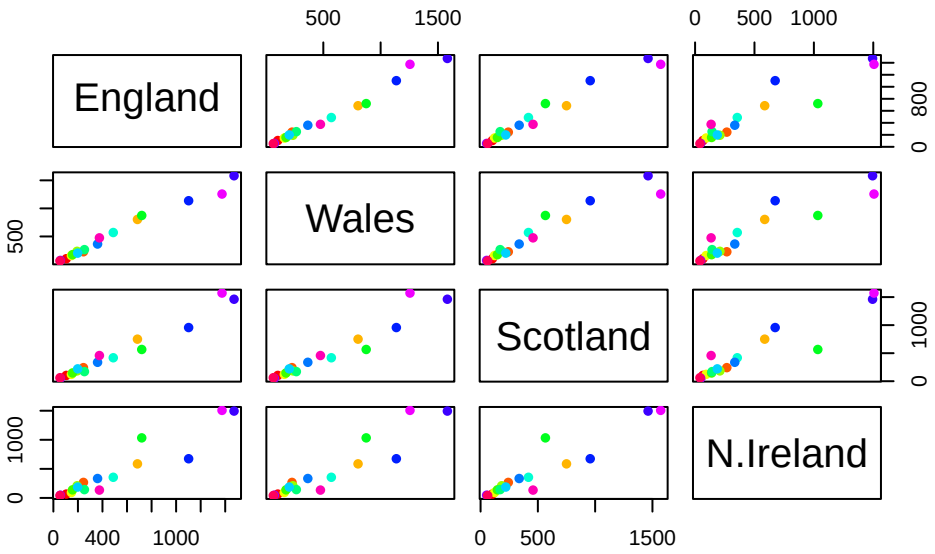

```
barplot(as.matrix(x), col = rainbow(nrow(x)), beside = T)
```



Q3: Changing what optional argument in the above **barplot()** function results in the following plot? **Changing `beside = F` will create the stacked bar plot.**

Pairwise plots for better comparison

```
pairs(x, col = rainbow(nrow(x)), pch = 16)
```



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? **Pairwise plots compare the different food types just within two countries hence it is easy to see the correlation between just 2 variables but not the entire dataset. If the point lies on the diagonal it shows how strongly correlated the density plots are.**

Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set? **N. Ireland has a more diverse distribution of food types when compared to the other countries as it does not have a clear diagonal. Instead there are more outliers and a wide spread of data points indicating weak correlation and therefore different ratio of food types consumed.**

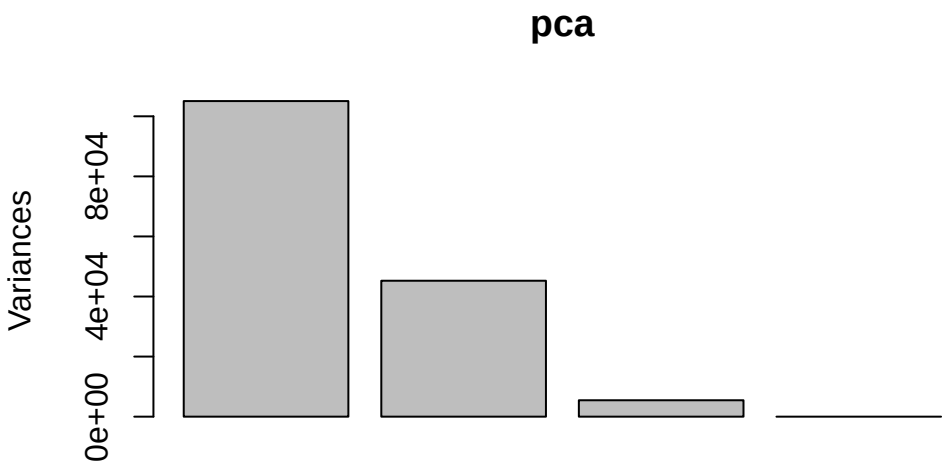
Let's apply PCA, for that we need to use the command `'prcomp()'`. This function expects the transpose of our data using `'t()'`.

```
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
```

Let's plot the PCA results

```
plot(pca)
```



We need to access the results of the PCA analysis

```
attributes(pca)
```

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

```
$class
[1] "prcomp"
```

We can explore the pca$x dataframe:

```
pca$x
```

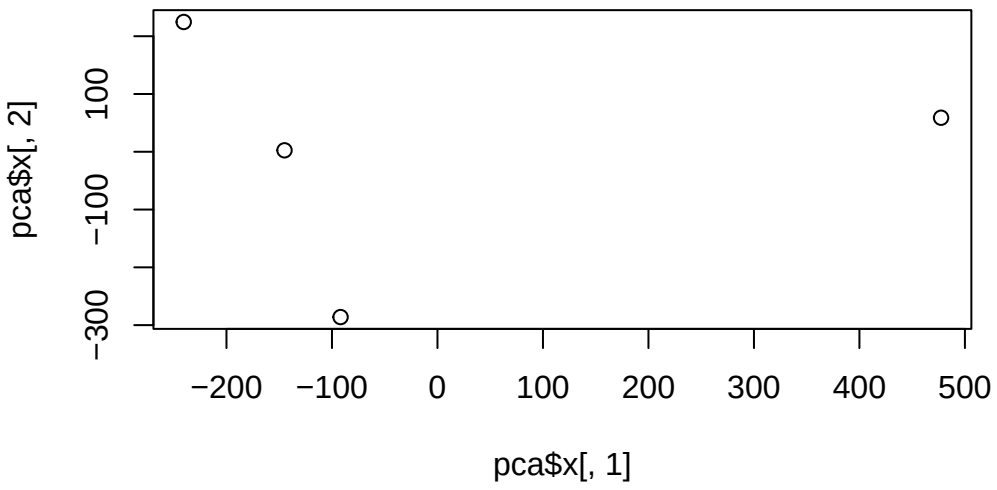
```
                PC1         PC2         PC3           PC4
England   -144.99315    2.532999 -105.768945  2.842865e-14
Wales     -240.52915  224.646925   56.475555  7.804382e-13
Scotland   -91.86934 -286.081786   44.415495 -9.614462e-13
N.Ireland  477.39164   58.901862    4.877895  1.448078e-13
```

```
pca$x[,1]
```

```
   England      Wales   Scotland  N.Ireland
-144.99315 -240.52915  -91.86934  477.39164
```

Plotting:

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

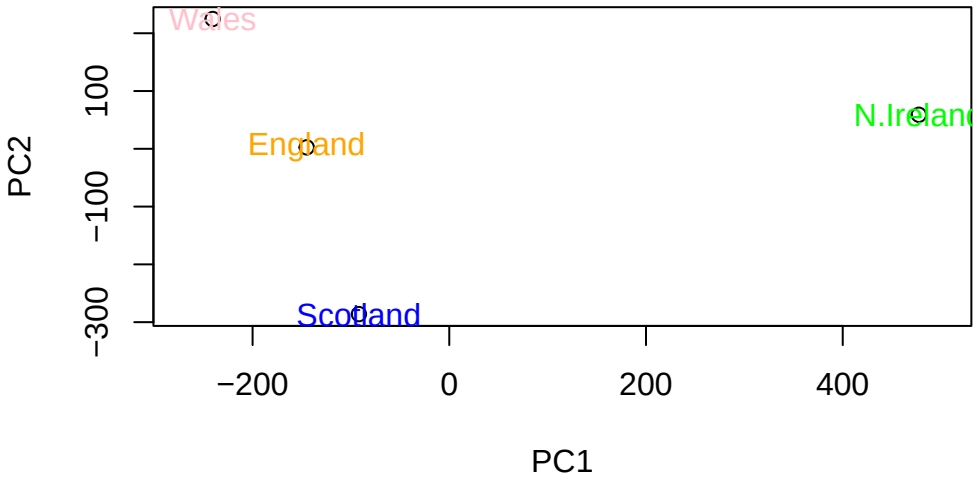


Plotting continued:

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

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.

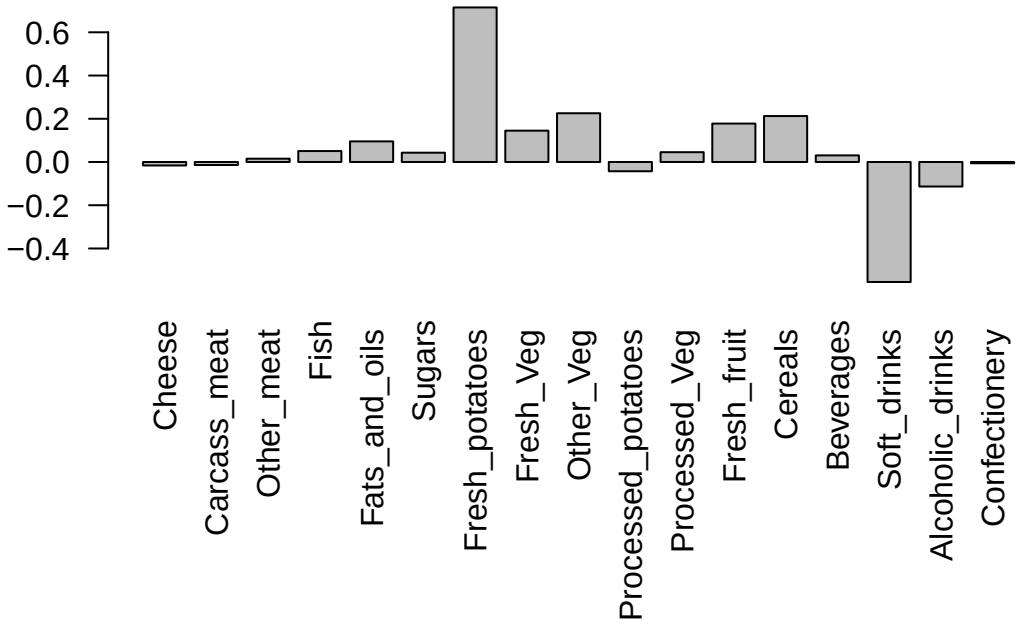
```
plot(x = pca$x[,1], y = pca$x[,2], xlab="PC1", ylab="PC2", xlim=c(-270,500))
colors_countries <- c('orange', 'pink', 'blue', 'green')
text(x = pca$x[,1], y = pca$x[,2], colnames(x), col = colors_countries)
```




```
# rownames(pca$x) is equivalent
```

Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantly and what does PC2 mainly tell us about? **PCA2 mainly tells us potatoes still have a very high positive loading score still and soft drinks has switched to a negative score.**

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



PCA of RNA Seq Data

First we import the 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? **100 genes and 10 samples**

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

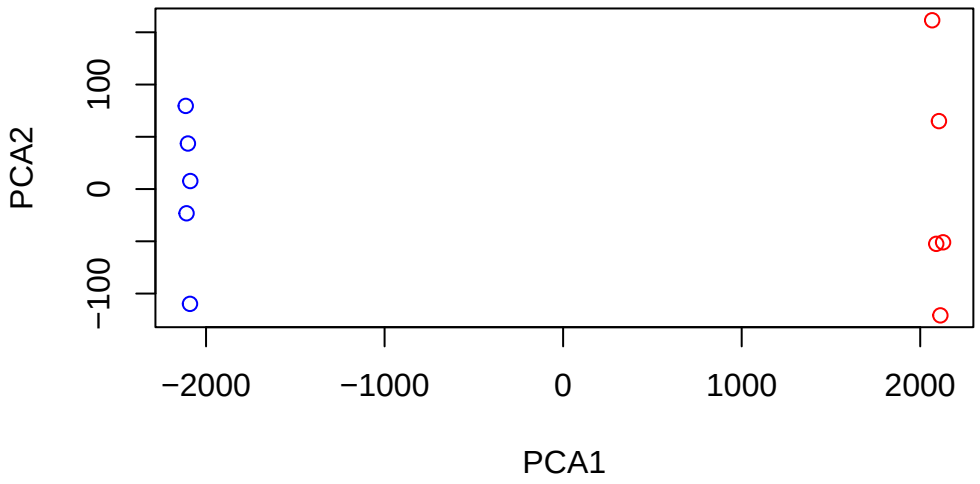
```
[1] 100  10
```

Applying PCA:

```
pca_rna <- prcomp(t(rna.data))
summary(pca_rna)
```

```
Importance of components:
                            PC1     PC2      PC3      PC4      PC5      PC6
Standard deviation     2214.2633 88.9209 84.33908 77.74094 69.66341 67.78516
Proportion of Variance    0.9917  0.0016  0.00144  0.00122  0.00098  0.00093
Cumulative Proportion     0.9917  0.9933  0.99471  0.99593  0.99691  0.99784
                           PC7      PC8      PC9      PC10
Standard deviation     65.29428 59.90981 53.20803 3.142e-13
Proportion of Variance  0.00086  0.00073  0.00057 0.000e+00
Cumulative Proportion   0.99870  0.99943  1.00000 1.000e+00
```

```
cols_samples <- c(rep('blue', 5), rep('red', 5))
plot(pca_rna$x[,1], pca_rna$x[,2],
     xlab = 'PCA1', ylab = 'PCA2',
     col = cols_samples)
```



Rotation shows expression levels

```r
barplot(pca_rna$rotation[,1])
```