W6_Machine_Learning_LAB

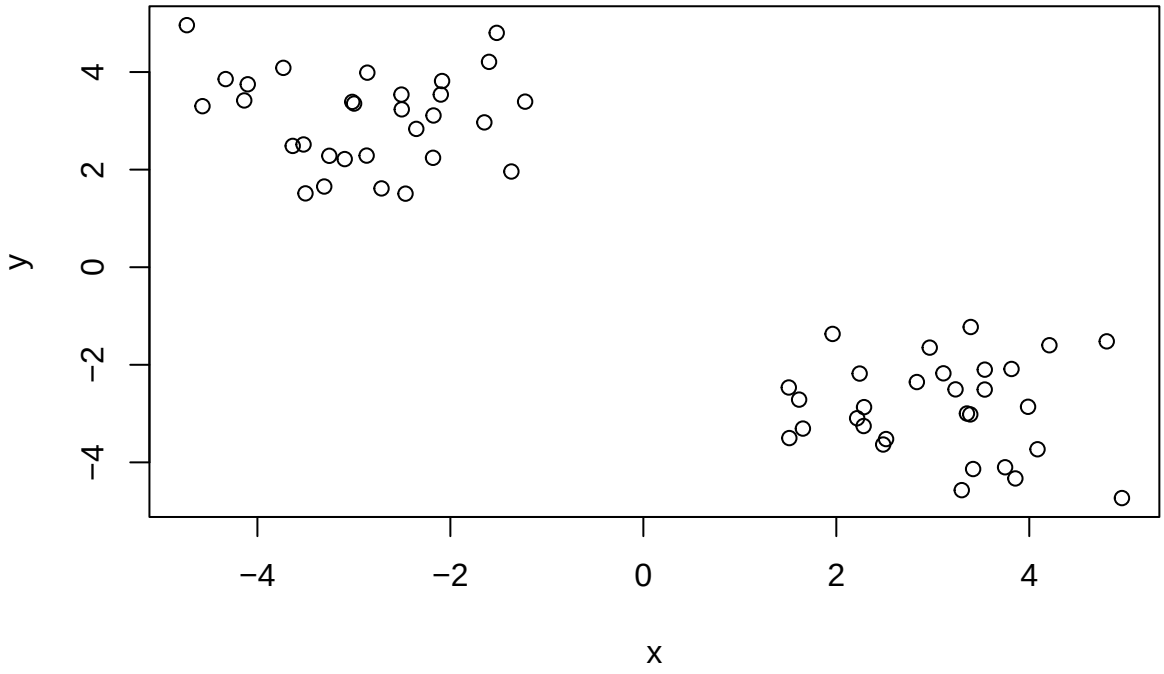
Vera Sophia Beliaev

2/12/2022

K Means

note to self: Alt+Ctrl+i is shortcut for inserting R code chunk In Kmeans, you impose structure onto data by defining the number of clusters

```
#Example data to cluster
tmp <- c(rnorm(30,-3), rnorm(30,3))
x <- cbind(x=tmp, y=rev(tmp))
plot(x)
```




```
#In kmeans, x is data input, centers is number of desired clusters, nstart is number of iterations)
k <- kmeans(x, centers=2, nstart=20)
k
```

```
## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##           x         y
## 1  3.060688 -2.869987
## 2 -2.869987  3.060688
##
## 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] 52.84231 52.84231
##  (between_SS / total_SS =  90.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Q. How many points are in each cluster?

```
k$size
```

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

Q. How do we get to the cluster assignment?

```
#Which cluster e/ data pnt is assigned to
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
```

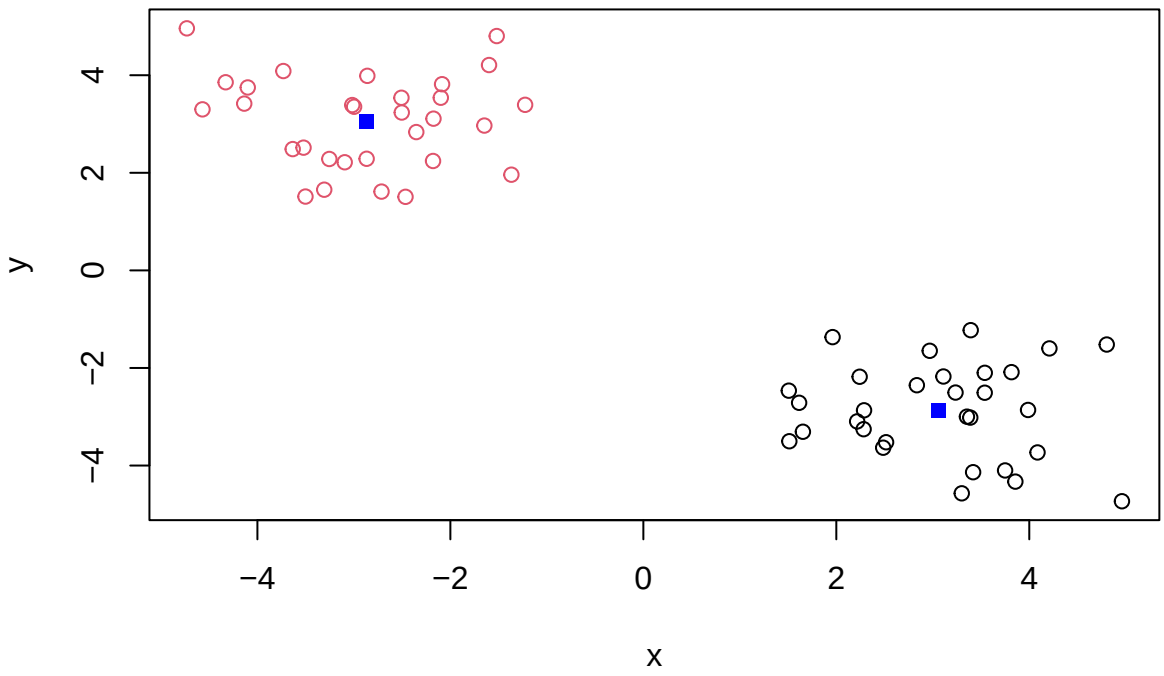
Q. Cluster centers?

```
k$centers
```

```
##           x         y
## 1  3.060688 -2.869987
## 2 -2.869987  3.060688
```

Application of results

```
#plot the kmeans by cluster assignment
plot(x, col=k$cluster)
# $ gives access to components of a list, pch= plotting character (shape)
points(k$centers, col= "blue", pch=15)
```



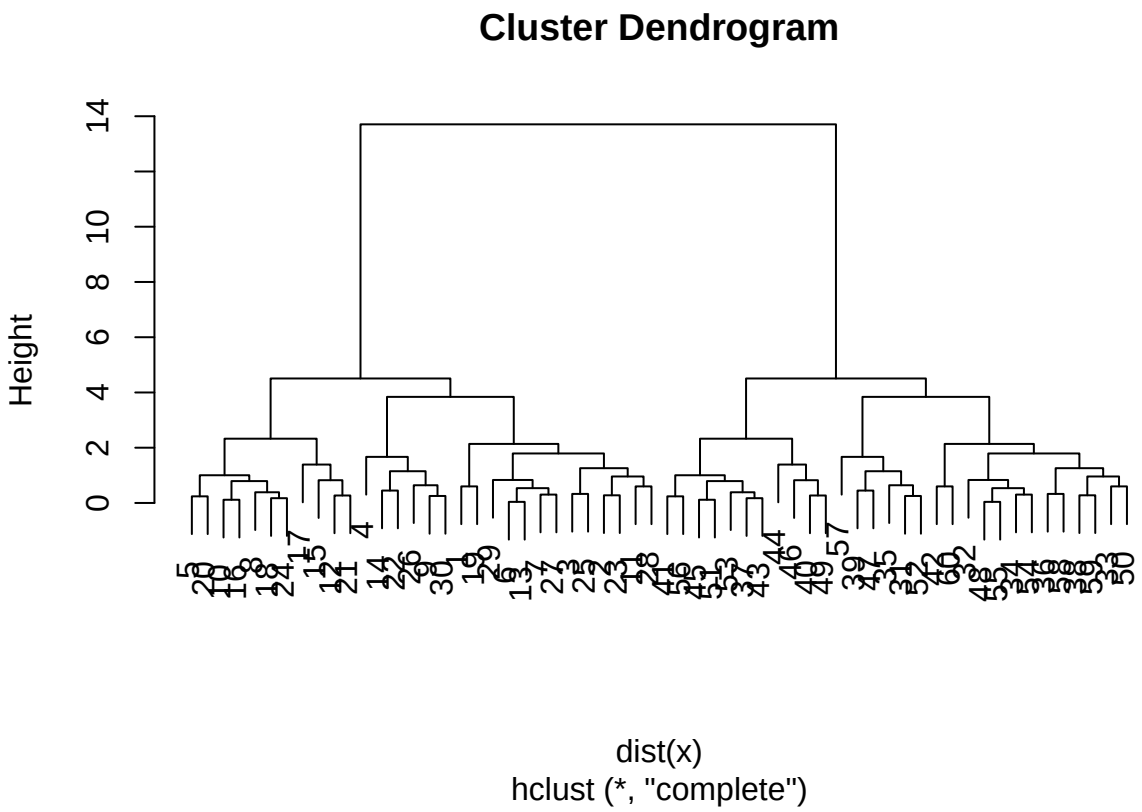
Hierarchical Clustering

Hierarchical clustering reveals structure in data Cluster the same data w/ hclust()

```
#hclust takes in a dissimilarity struc, not the data, given by distance matrix
hc <- hclust(dist(x))
hc
```

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

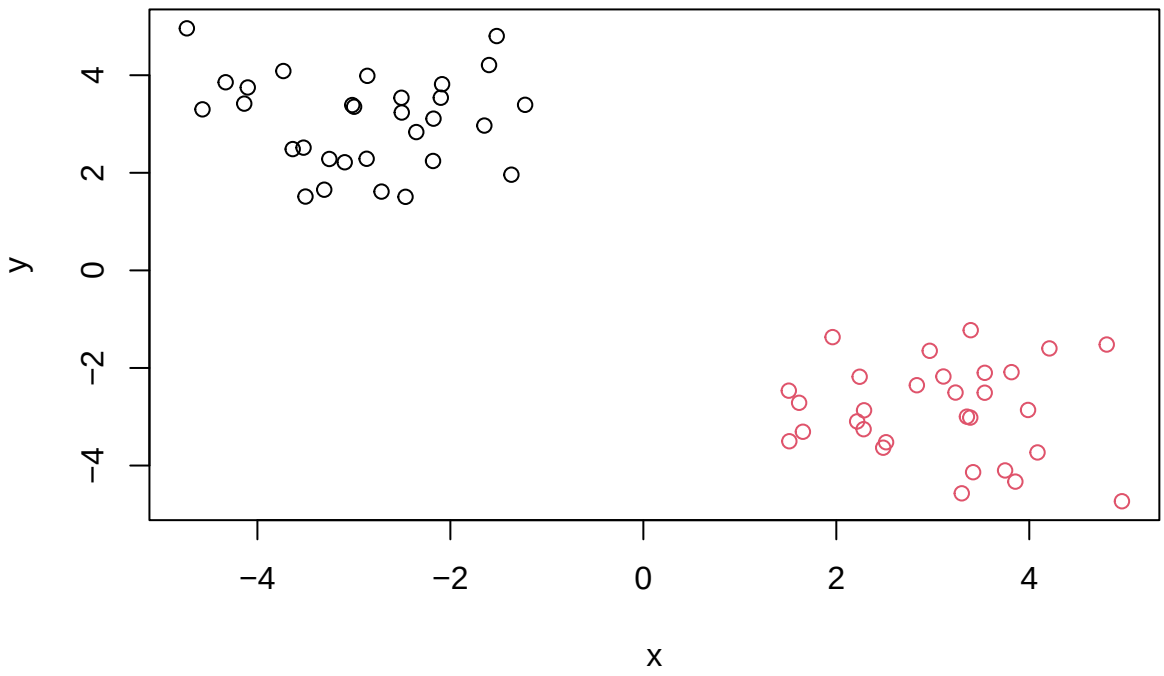
```
#Plot the hclust result
plot(hc)
```



You need to "cut" the tree to figure out cluster membership, use "cutree()"

```
#have to give it a height/h to cut across at,base it on the plot
#this gives us the cluster membership vector
#can also do k= to cut into the # of groups you want
grps <- cutree(hc, h=8)
```

```
#Plot the data w/ the hclust() results, color according to group membership
plot(x, col=grps)
```



Principal Component Analysis (PCA)

PCA of UK food data

```
url <- "https://tinyurl.com/UK-foods"
#row.names.1 will make it take the row names from the data in our first column
x <- read.csv(url, row.names=1)
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
## Fresh_potatoes          720   874      566      1033
## Fresh_Veg               253   265      171       143
## Other_Veg               488   570      418       355
## Processed_potatoes      198   203      220       187
## Processed_Veg           360   365      337       334
## Fresh_fruit            1102  1137      957       674
## Cereals                1472  1582     1462      1494
## Beverages                57    73       53        47
## Soft_drinks            1374  1256     1572      1506
## Alcoholic_drinks        375   475      458       135
## 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?

```
ncol(x)
```

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

```
nrow(x)
```

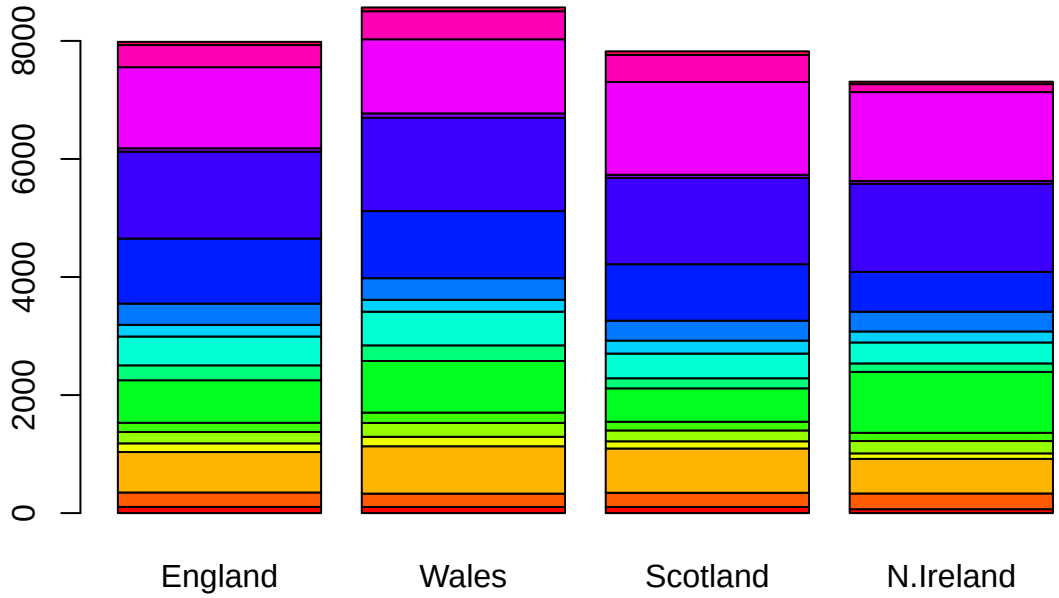
```
## [1] 17
```

```
#alternatively, you can use dim() to return both
```

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?

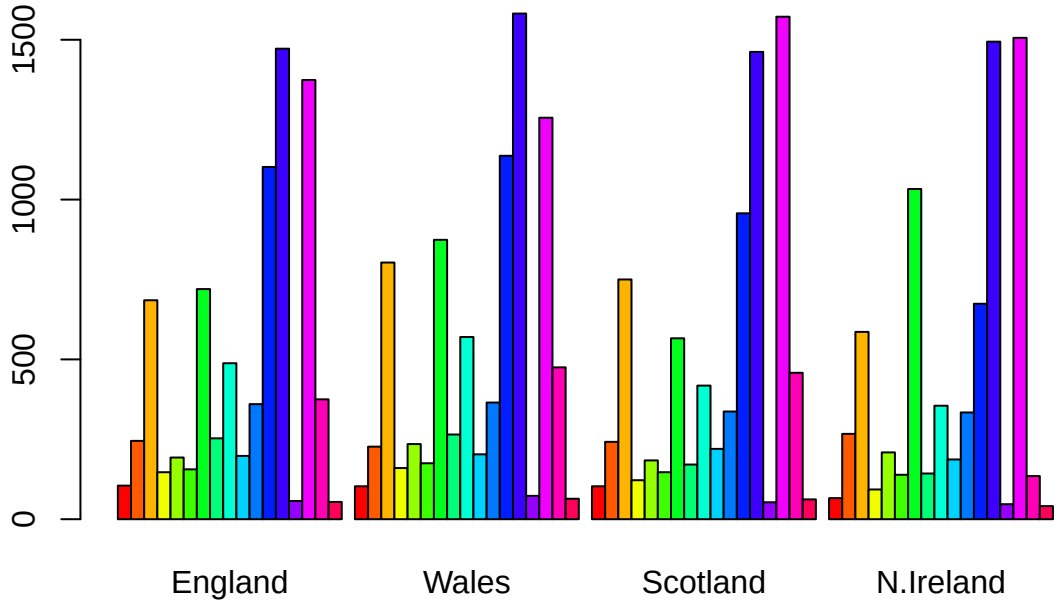
The row.names=1 approach as an arg to read.csv seems most efficient

```
#specify number of rainbow colors, here we base it on number of rows
barplot(as.matrix(x), col= rainbow(nrow(x)))
```



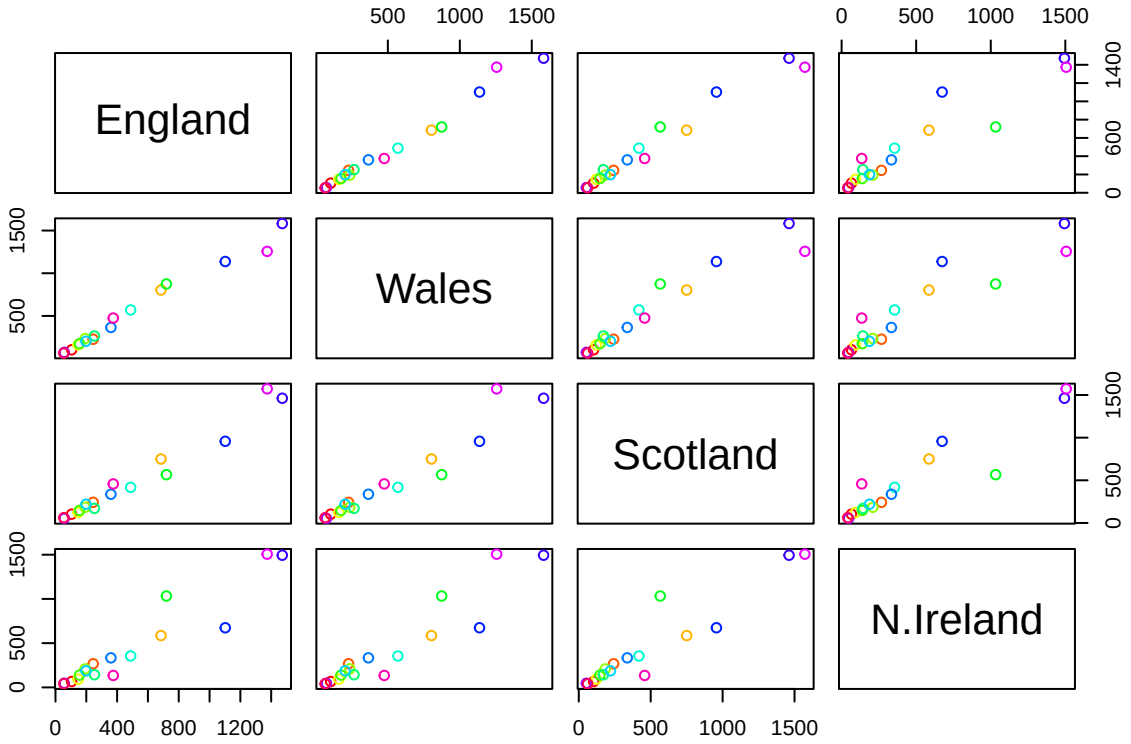
>Q3. Changing what optional argument in the below barplot() function results in the following plot? Changing the "beside=" argument between TRUE/FALSE switches between the two bar plots.

```
#beside=T will give juxtaposed bars, default false stacks
barplot(as.matrix(x), col= rainbow(nrow(x)), beside=TRUE)
```



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? The resulting figure compares two countries at a time in each plot. Lying on the diagnol means there is not significant difference between the two.

```
#pairs gives all possible plots of column vs column
pairs(x, col= rainbow(nrow(x)))
```



Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set? It is hard to tell in which categories but there are some differences i.e. things are not lined up on the diagnol. Potatos and alcohol?

PCA to the rescue

THe main R PCA fn is prcomp(), it reqs the transpose of your input data

```
# t(x) is the transpose of the data which means now the columns are rows and the rows are columns, in t
#for prcomp, you need to use the transpose of the data
pca <- prcomp(t(x))
```

```
#this hows how well PCA is doing, proportion of variance shows how much variance is captured in e/ PC
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
```

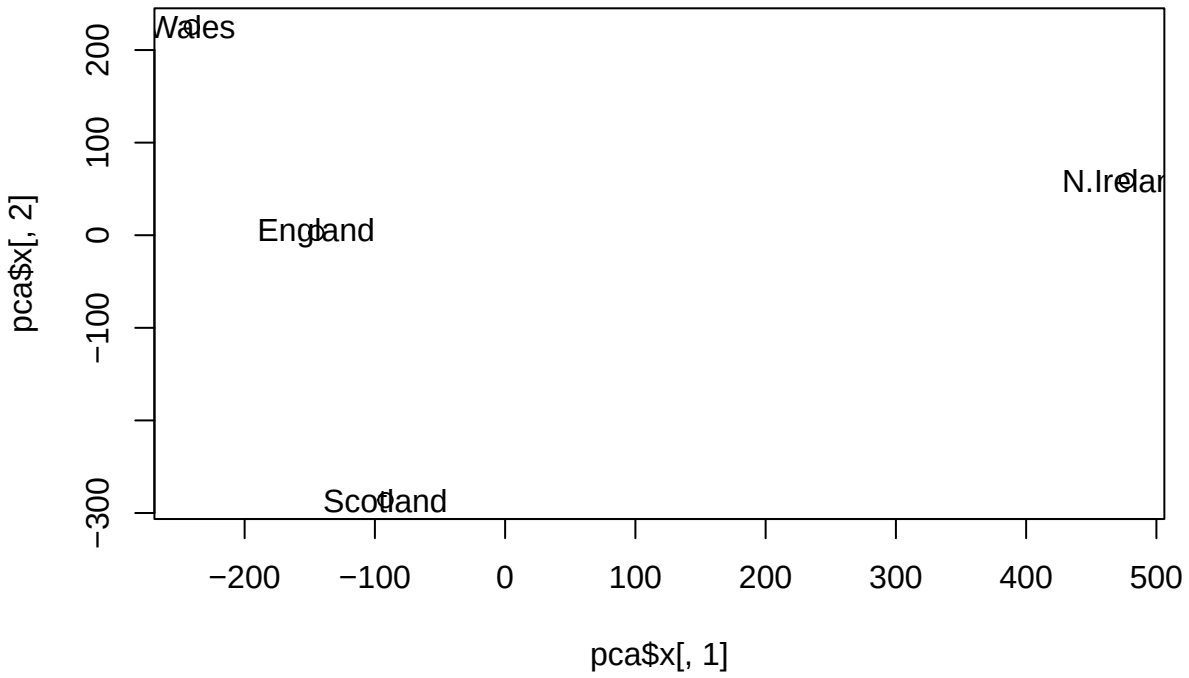
```
## [1] "sdev"     "rotation" "center"   "scale"    "x"
##
## $class
## [1] "prcomp"
```

Make new PCA plot (aka PCA score plot), we have to access pca$x

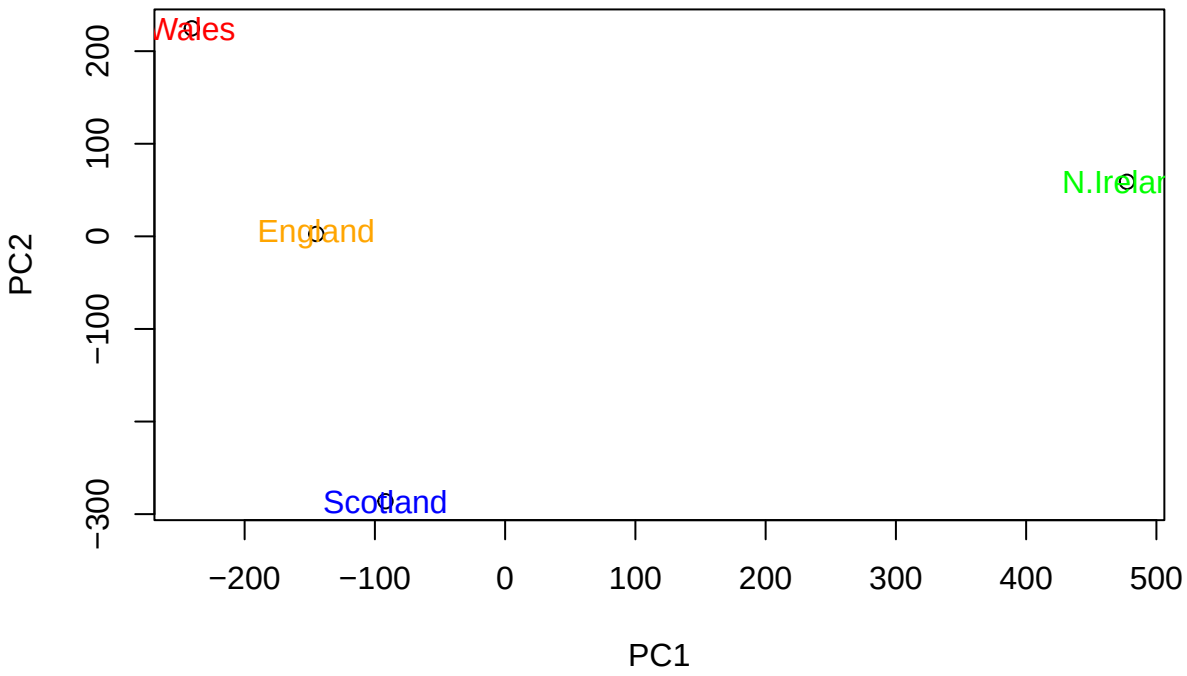
```
pca$x
```

```
##                  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
```

```
#we want to plot PC1 vs PC2
plot(pca$x[,1], pca$x[,2])
text(pca$x[,1], pca$x[,2], colnames(x))
```

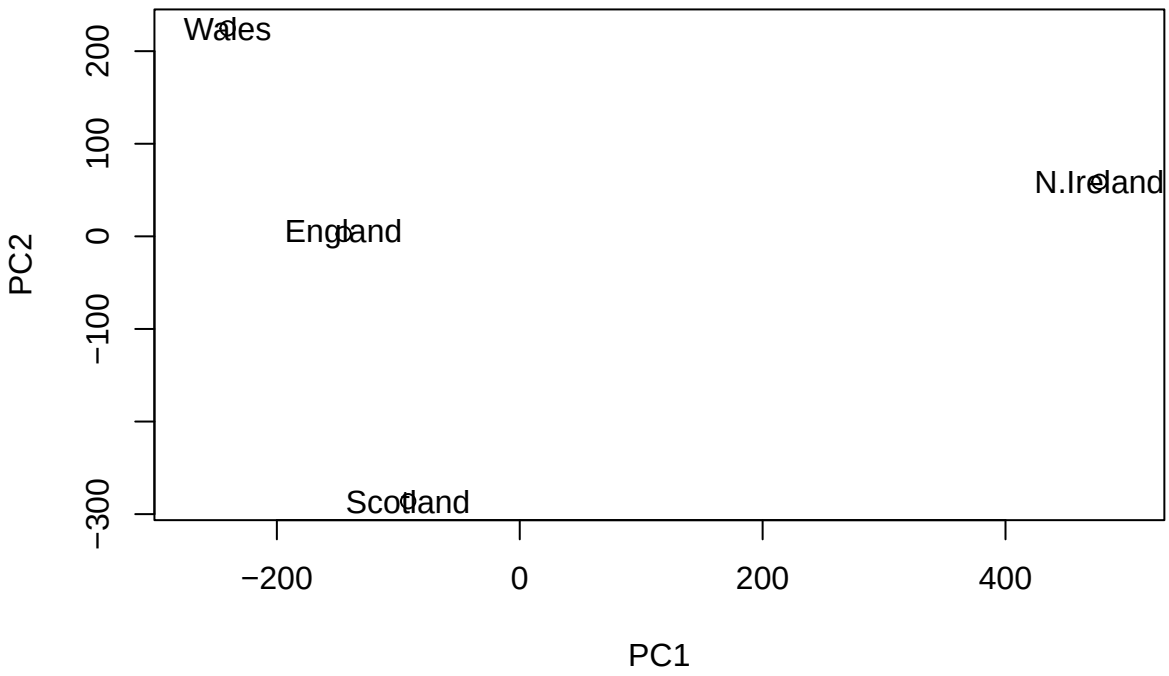



```
#add color to plot
country_cols <- c("orange", "red", "blue", "green")
plot(pca$x[,1], pca$x[,2], xlab= "PC1", ylab = "PC2")
text(pca$x[,1], pca$x[,2], colnames(x), col=country_cols)
```



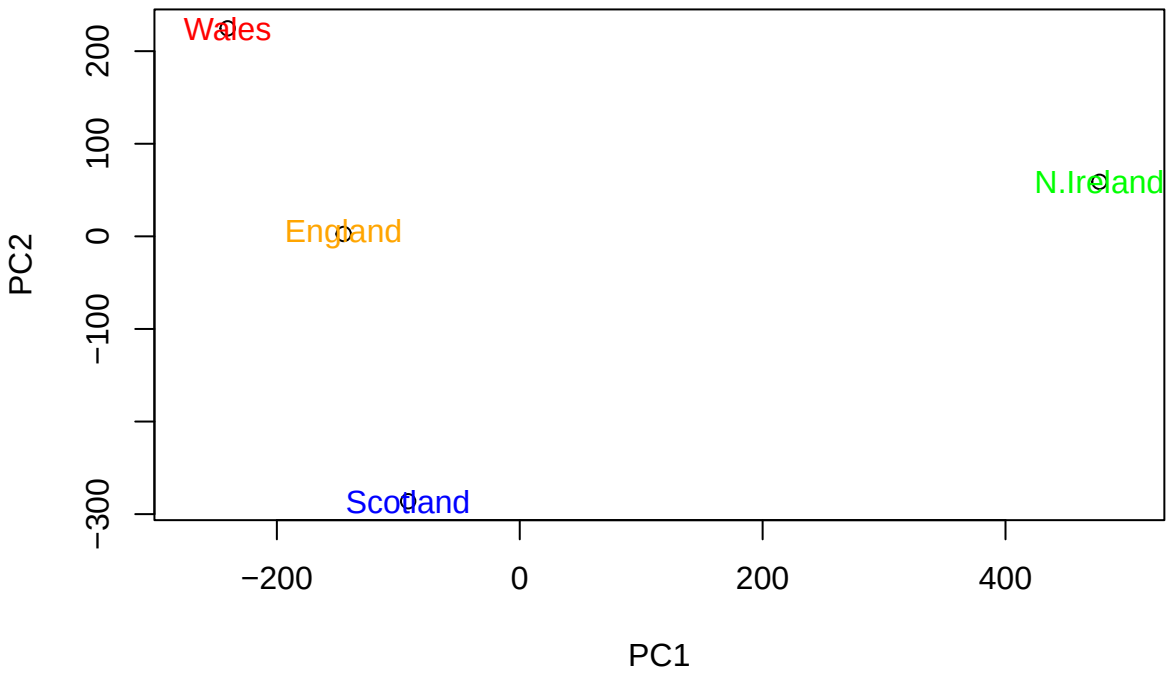
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
# 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))
```



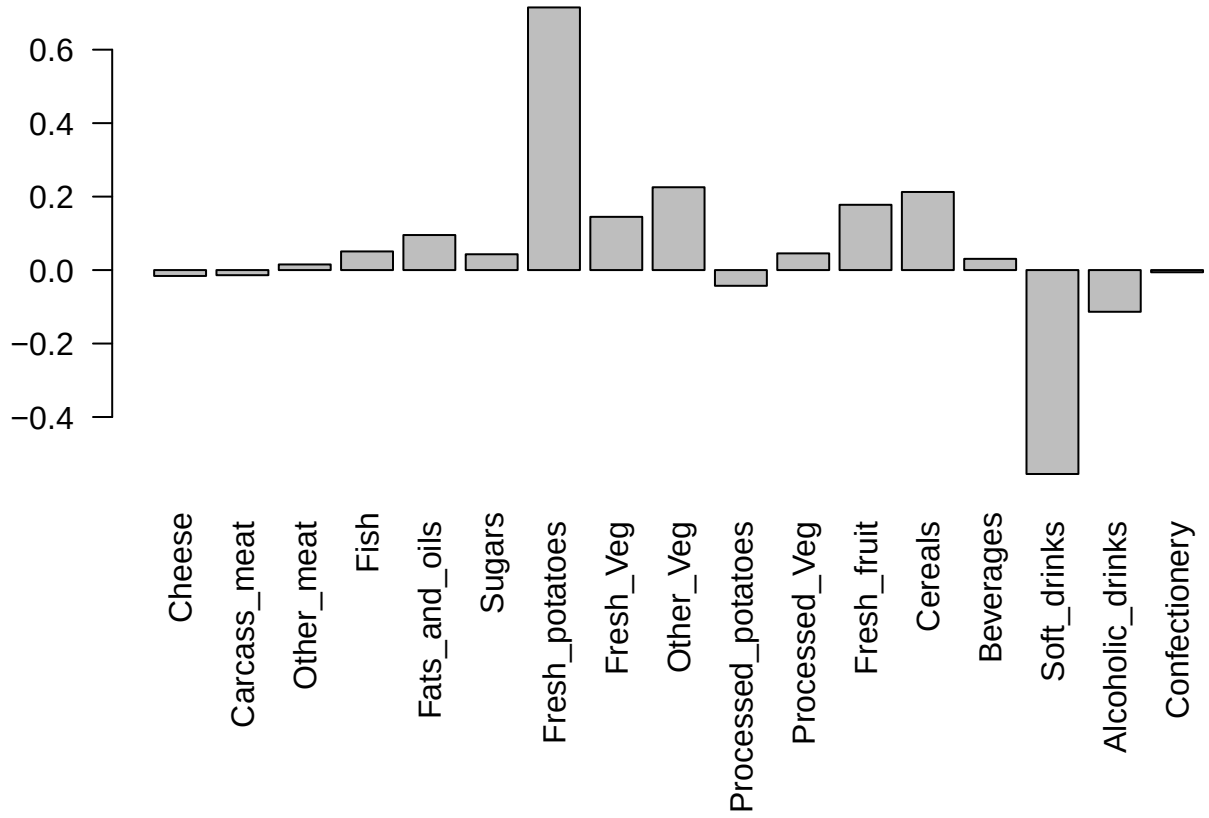
Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
country_cols <- c("orange", "red", "blue", "green")
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=country_cols)
```



Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about? The biggest contributers to PC2 and are the fresh potatoes and soft drinks food groups.

```
## 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 )
```



PCA of RNA-seq data

```
#always use read.csv for data files
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? There are 100 genes and 10 samples

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

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

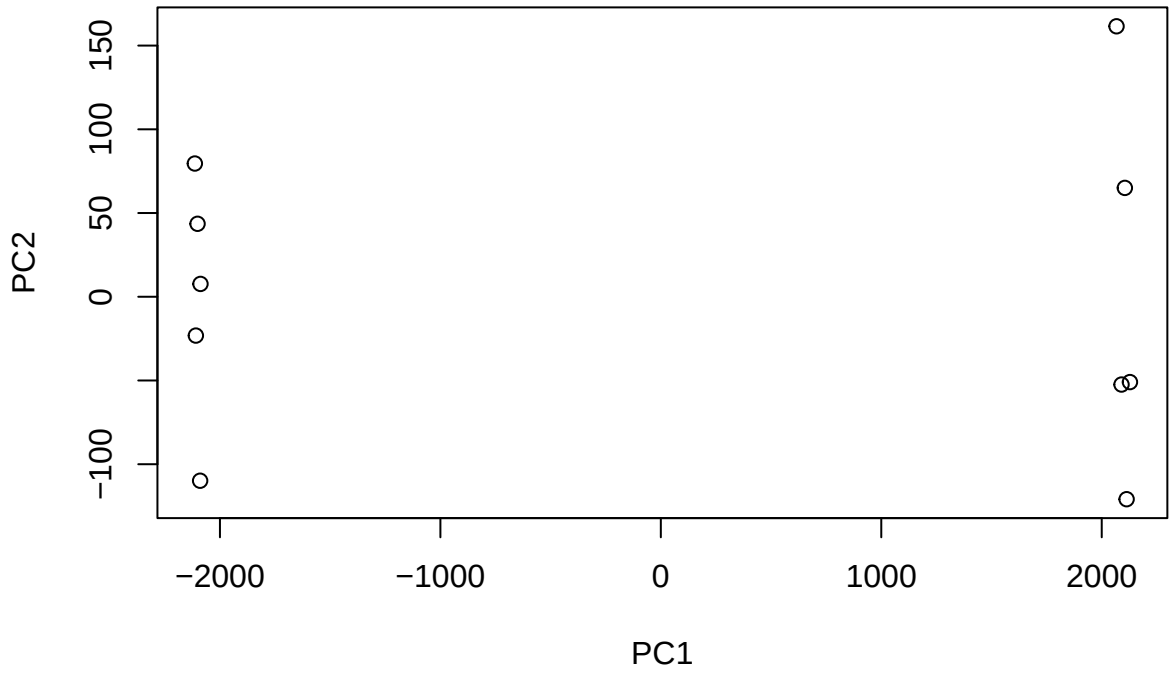
Make PCA plot

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

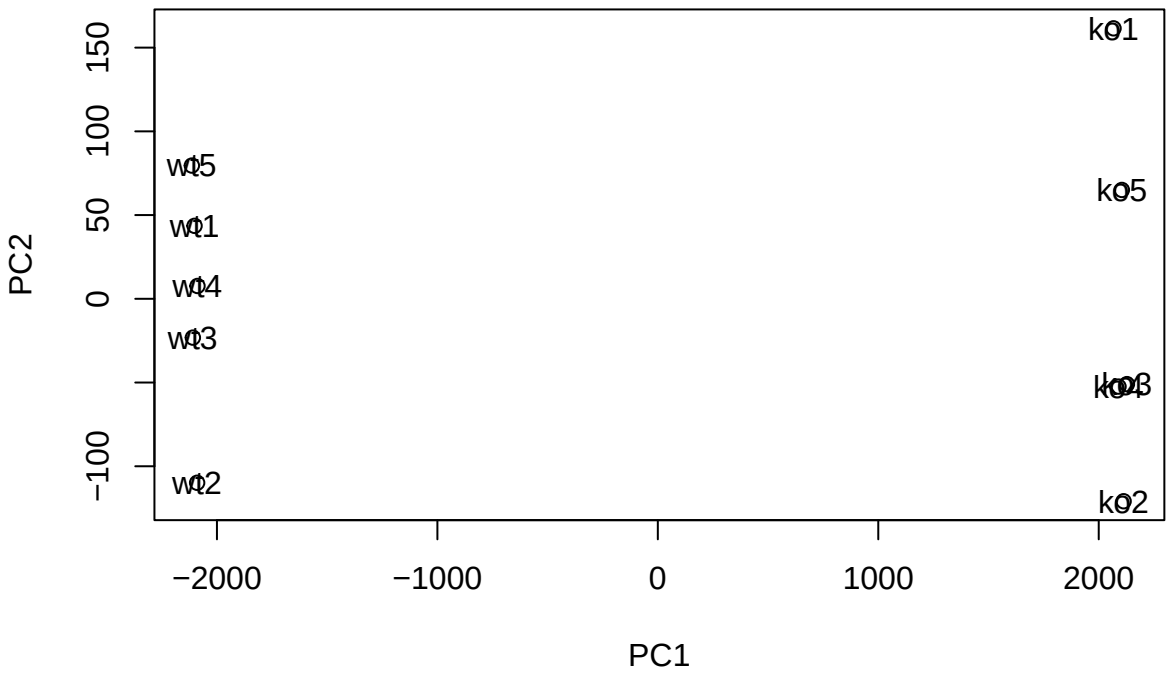
```
## Importance of components:
##                             PC1     PC2      PC3      PC4      PC5      PC6
## Standard deviation     2214.2633 88.9209 84.33908 77.74094 69.66341 67.78516
## Proportion of Variance    0.9917  0.0016  0.00144  0.00122  0.00098  0.00093
## Cumulative Proportion     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
```

Most of variance is captured in PC1

```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab= "PC2")
```




```
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab= "PC2")
text(pca$x[,1], pca$x[,2], colnames(rna.data))
```


PC2
150
100
50
0
−100
wt5
wt1
wt4
wt3
wt2
ko1
ko5
ko4
ko3
ko2
−2000
−1000
0
1000
2000
PC1