Machine Learning 1

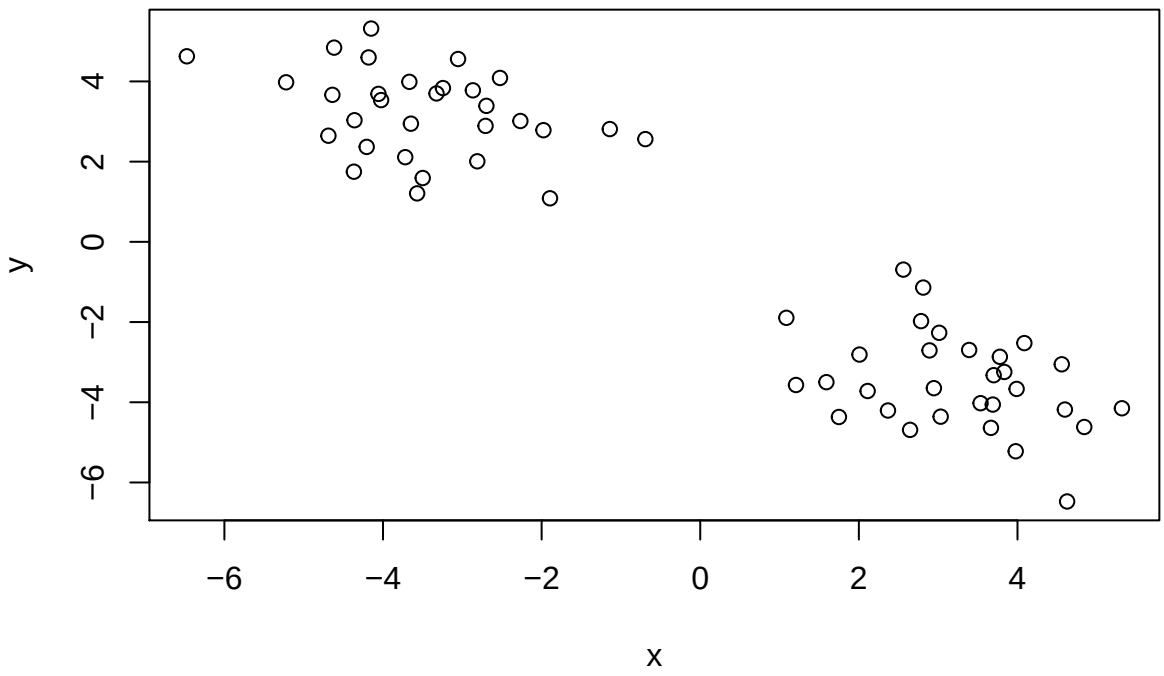
Soobin (PID:A15201229)

2/10/2022

First up kmeans()

Demo of using kmeans() function in base R. First make up some data with a known structure.

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



Now we have some made up data in 'x' let's see how kmeans works with this data

```
# k means algorithm with 2 centers and run 20 times
k <- kmeans(x, centers = 2, nstart = 20)
k
```

```
## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##           x         y
## 1 -3.476284  3.212848
## 2  3.212848 -3.476284
##
## 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 1 2 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] 76.87773 76.87773
##  (between_SS / total_SS =  89.7 %)
##
## 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 cluster membership/assignment?

```
k$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 1 2 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 about cluster centers?

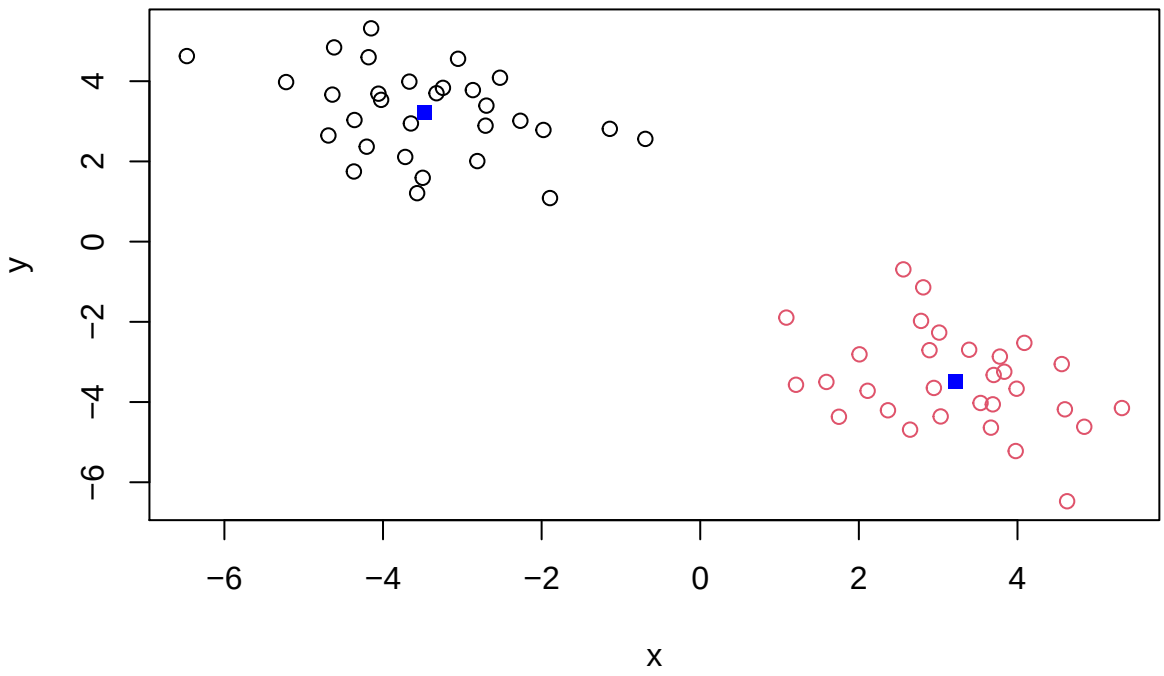
```
k$centers
```

```
##           x         y
## 1 -3.476284  3.212848
## 2  3.212848 -3.476284
```

Now we got to the main results let's use them to plot our data with the kmeans result

```
plot(x, col=k$cluster)

# cluster centers
points(k$centers, col="blue", pch=15)
```



Now for Hierarchical Clustering

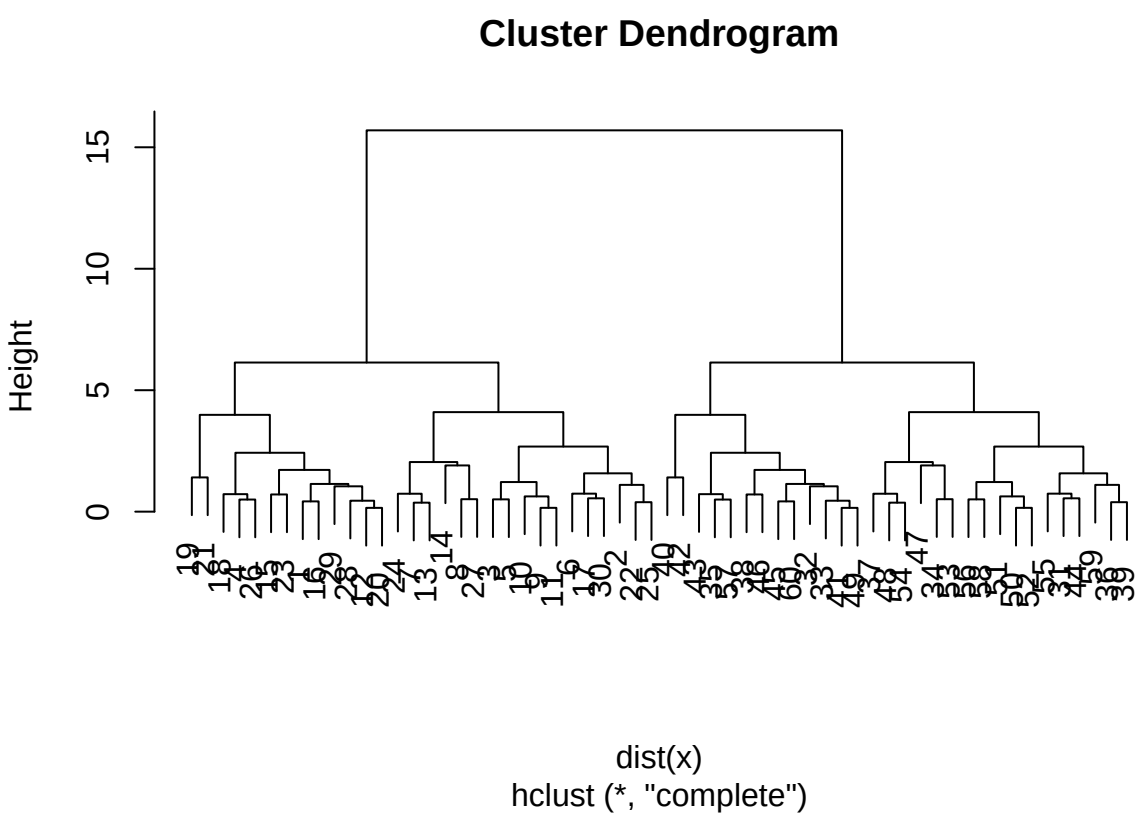
We will cluster the same data 'x' with the 'hclust()'. In this case, 'hclust()' requires a distance matrix as input.

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

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

Let's plot our hclust result.

```
plot(hc)
```



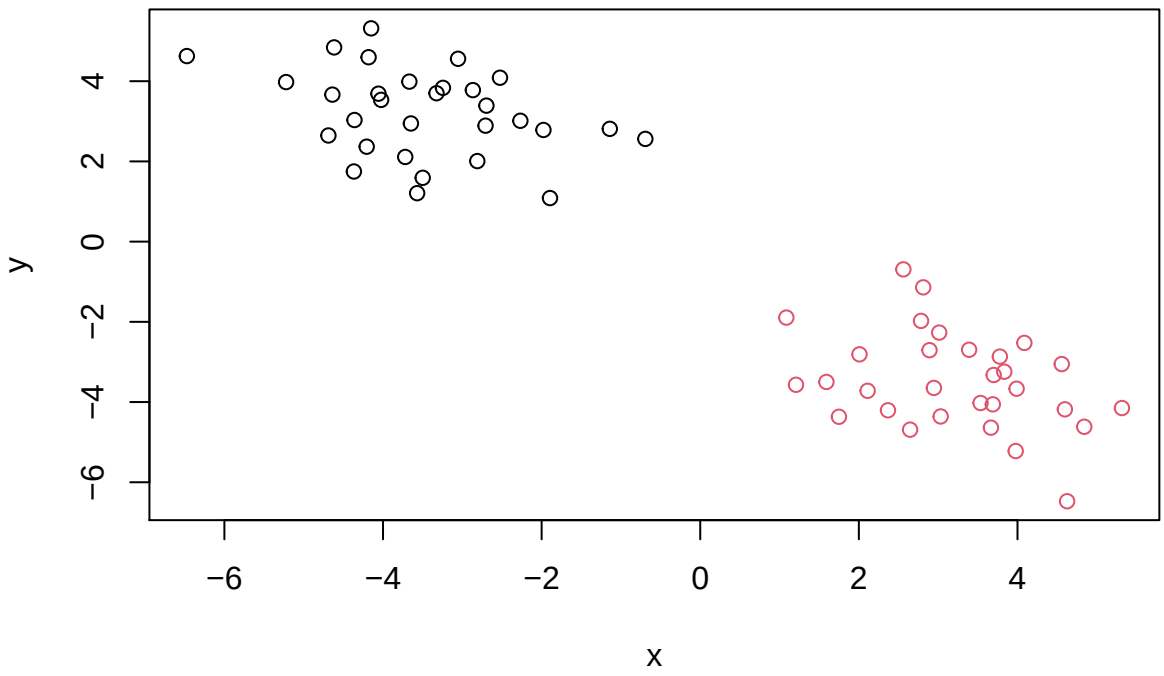
To get our cluster membership vector we need to "cut" the tree with the 'cutree()'.

```
grps <- cutree(hc, h=8)
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 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
```

Now plot our data with the hclust() results.

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



Principal Component Analysis (PCA)

PCA of UK food data

Read data from website and try a few visualizations.

```
url <- "https://tinyurl.com/UK-foods"
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?

There is 17 rows and 4 columns. There would be 5 columns if you did not state 'row.names=1'. I would use dim function to answer this question.

```
dim(x)
```

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

Checking your data

```
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
```

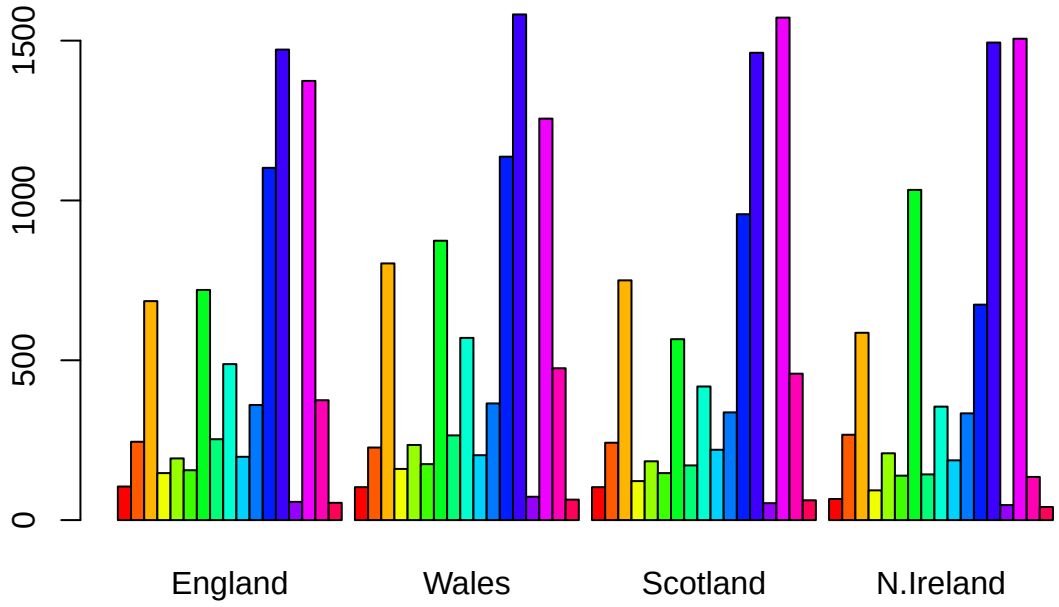
> Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

I prefer to set 'row.names=1' of read.csv() rather than following the method below. The method below will constantly reset the first column to take the row names so we will lose our data everytime the following code is run.

```
# rownames(x) <- x[,1]
# x <- x[,-1]
# head(x)
```

Spotting major differences and trends

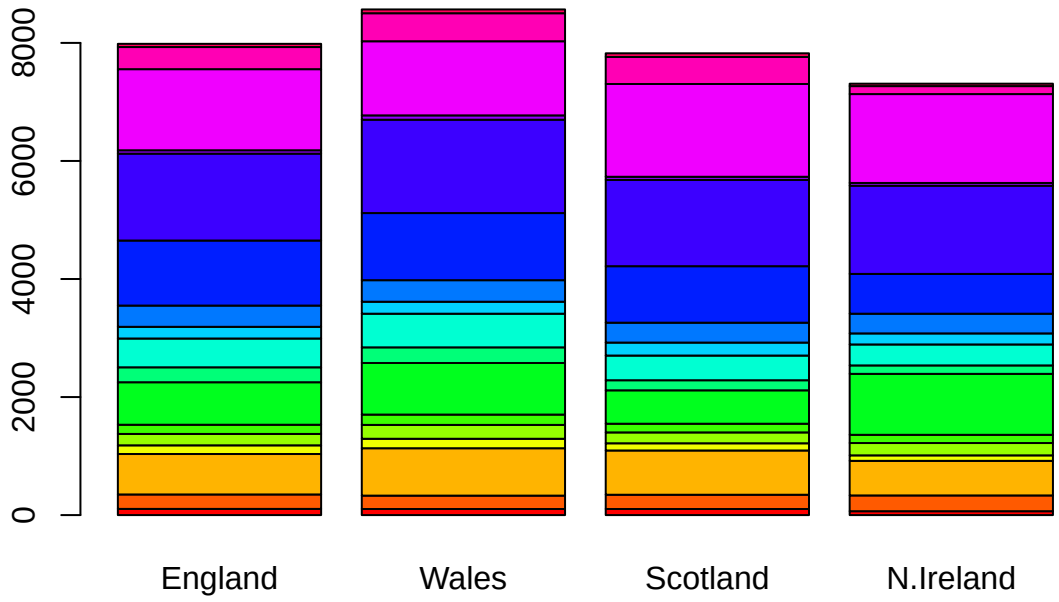
```
cols <- rainbow( nrow(x) )
barplot( as.matrix(x), col=cols, beside=TRUE )
```



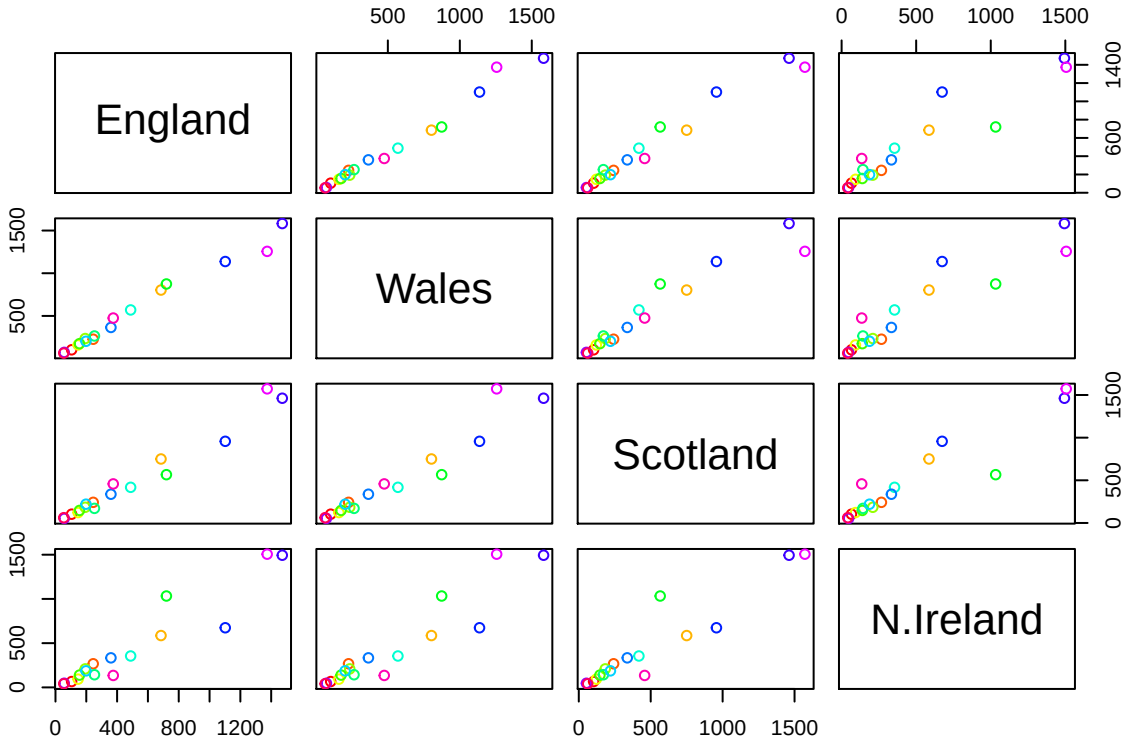
Q3: Changing what optional argument in the above barplot() function results in the following plot?

Chaning 'besides=T' of barplot() to 'besides=F' results in the following plot.

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




```
pairs(x, col=cols)
```



(no Q4) 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?

Each plot is comparing the food consumption between two countries. Depending on the location of the pairwise plot, the x-axis and the y-axis can change. If a point lies above the diagonal, this means that the point (specific food) is consumed more in the y-axis country than in the x-axis country. This is vise versa when the point lies below the diagonal.

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

The blue point and the green point are the main differences between N. Ireland and other countries of the UK. The blue point (likely fresh potatoes) indicate that N. Ireland consumes more than other countries, while the green point (likely alcoholic drinks) indicate that N. Ireland consumes less than other countries.

PCA to the rescue!!

The main base R PCA function is called 'prcomp()' and we will need it the transpose of our input data!

```
pca <-  prcomp( t(x) )
pca
```

```
## Standard deviations (1, .., p=4):
## [1] 3.241502e+02 2.127478e+02 7.387622e+01 2.921348e-14
##
## Rotation (n x k) = (17 x 4):
##                              PC1          PC2         PC3          PC4
## Cheese               -0.056955380 -0.016012850 -0.02394295 -0.409382587
## Carcass_meat          0.047927628 -0.013915823 -0.06367111  0.729481922
## Other_meat           -0.258916658  0.015331138  0.55384854  0.331001134
## Fish                 -0.084414983  0.050754947 -0.03906481  0.022375878
## Fats_and_oils        -0.005193623  0.095388656  0.12522257  0.034512161
## Sugars               -0.037620983  0.043021699  0.03605745  0.024943337
## Fresh_potatoes        0.401402060  0.715017078  0.20668248  0.021396007
## Fresh_Veg            -0.151849942  0.144900268 -0.21382237  0.001606882
## Other_Veg            -0.243593729  0.225450923  0.05332841  0.031153231
## Processed_potatoes   -0.026886233 -0.042850761  0.07364902 -0.017379680
## Processed_Veg        -0.036488269  0.045451802 -0.05289191  0.021250980
## Fresh_fruit          -0.632640898  0.177740743 -0.40012865  0.227657348
## Cereals              -0.047702858  0.212599678  0.35884921  0.100043319
## Beverages            -0.026187756  0.030560542  0.04135860 -0.018382072
## Soft_drinks           0.232244140 -0.555124311  0.16942648  0.222319484
## Alcoholic_drinks     -0.463968168 -0.113536523  0.49858320 -0.273126013
## Confectionery        -0.029650201 -0.005949921  0.05232164  0.001890737
```

There is a nice summary of how well PCA is doing.

```
summary(pca)
```

```
## Importance of components:
##                             PC1      PC2      PC3       PC4
## Standard deviation     324.1502 212.7478 73.87622 2.921e-14
## Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
## Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

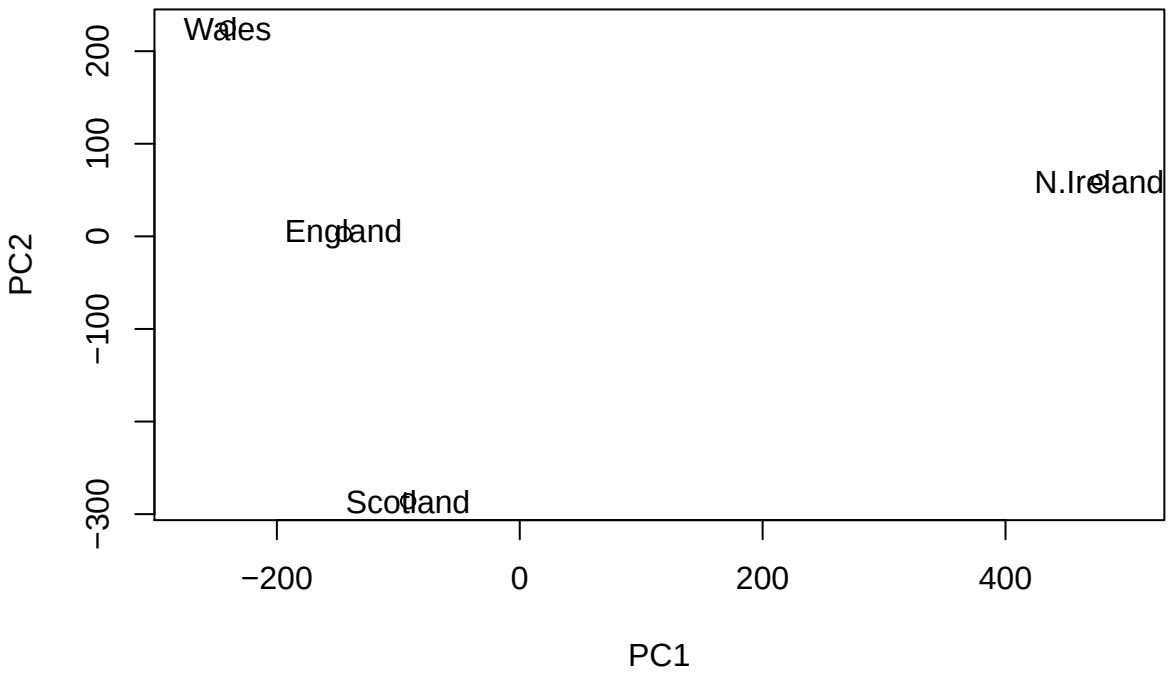
```
attributes(pca)
```

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

To make our new PCA plot (a.k.a. PCA score plot) we access 'pca$x'.

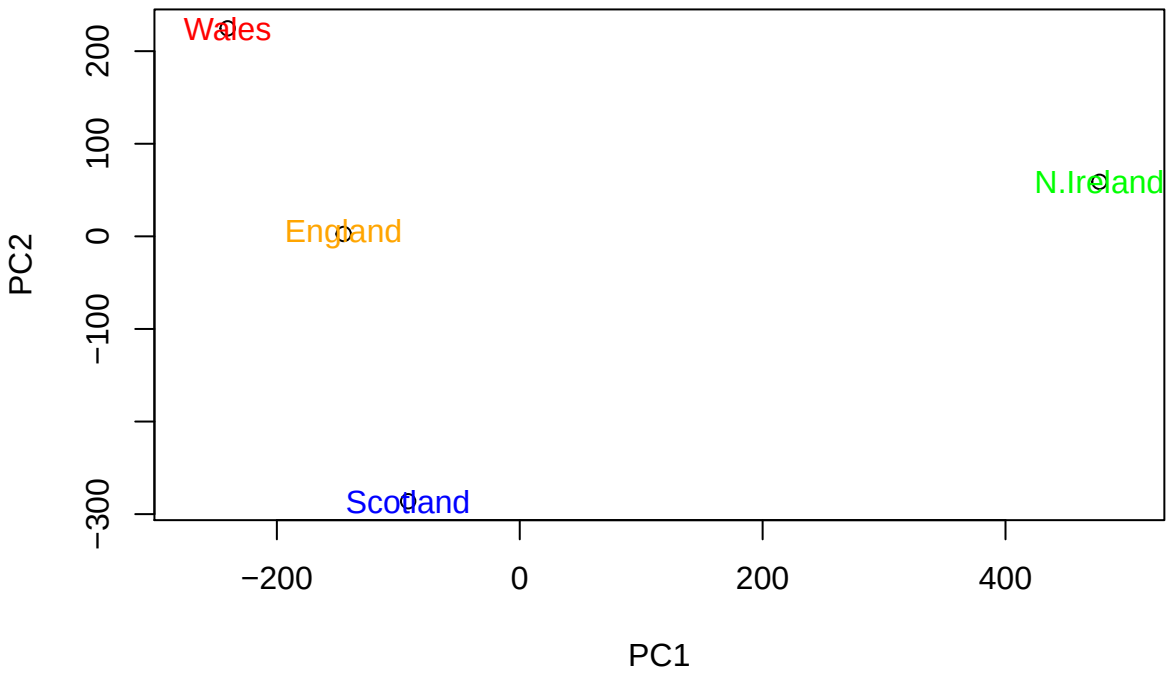
> Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
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.

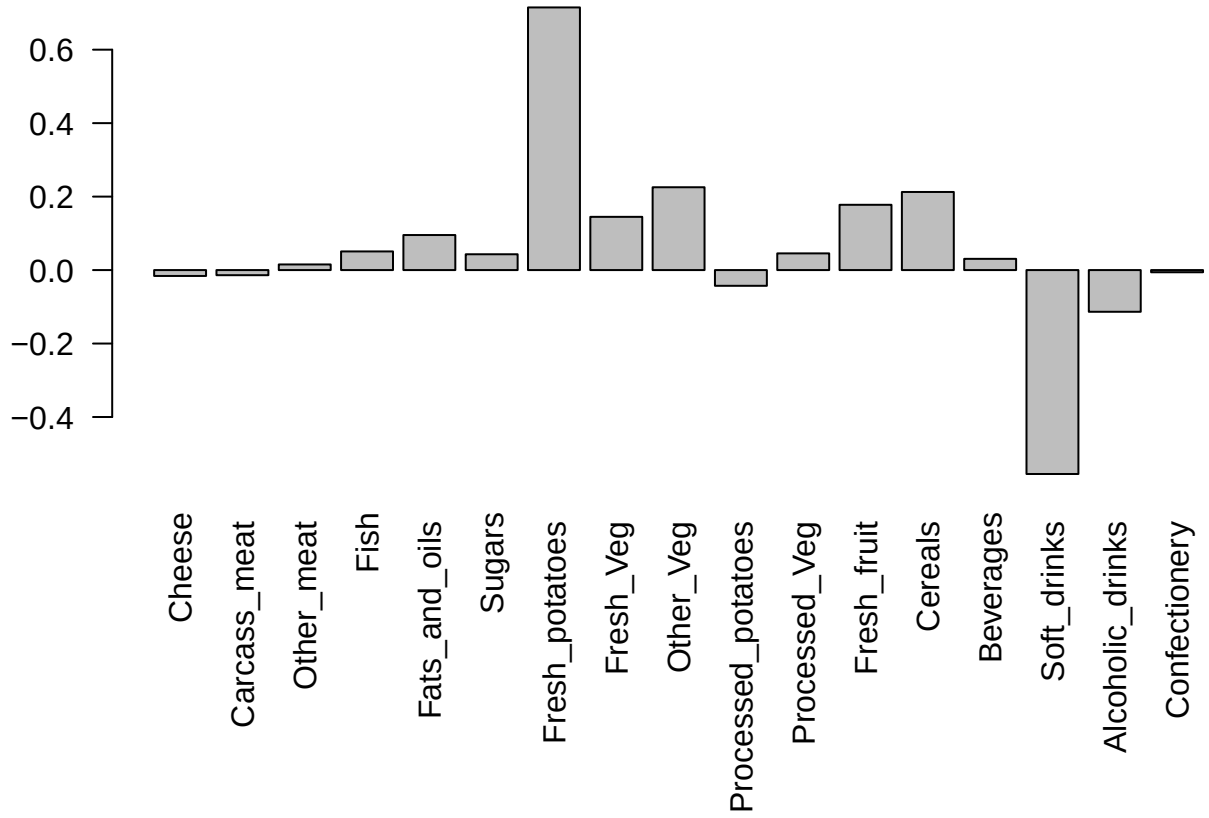
```
country_cols <- c("orange", "red", "blue", "green")
plot( pca$x[, 1], pca$x[, 2], xlab = "PC1", ylab = "PC2", xlim=c(-270,500) )
text( pca$x[, 1], pca$x[, 2], colnames(x), col=country_cols)
```



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

Soft_drink has the largest negative score, and fresh_potatoes has the largest positive loading scores. Fresh_potatoes and soft_drink contribute the most in PC2 variance.

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



PCA of RNA-seq data

Read in data from website

```
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 in this data set.

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

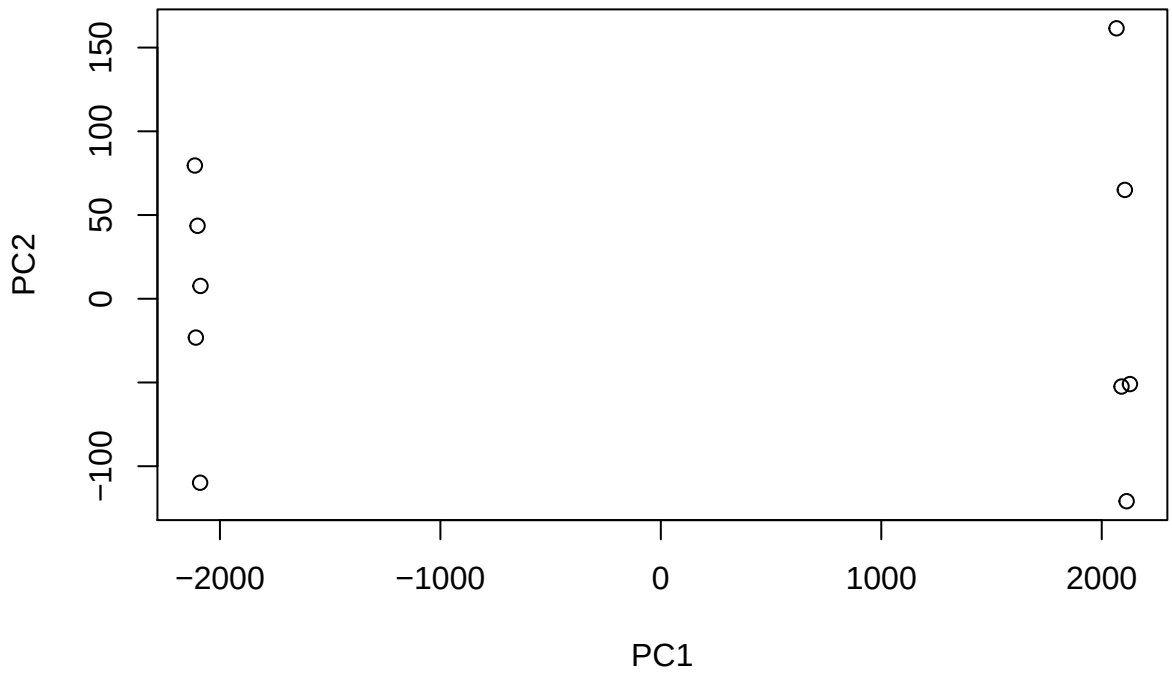
```
## [1] 100  10
```

```
pca1 <- prcomp( t(rna.data) )
summary(pca1)
```

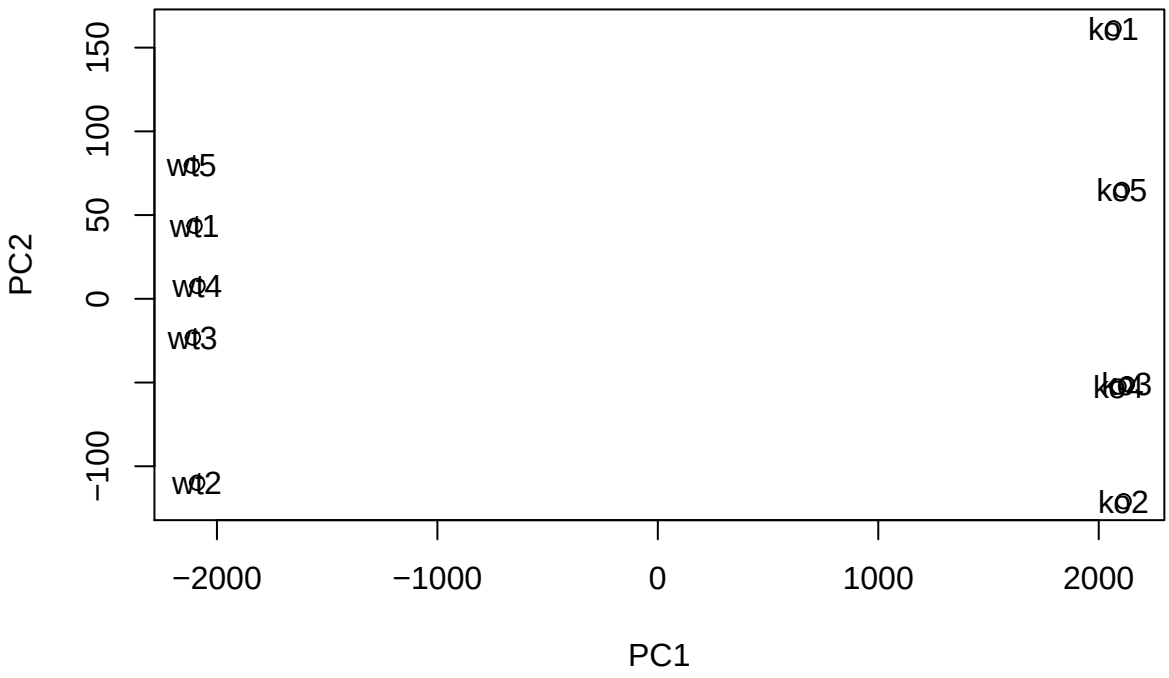
```
## Importance of components:
##                             PC1     PC2      PC3      PC4      PC5      PC6
## Standard deviation     2214.2633 88.9209 84.33908 77.74094 69.66341 67.78516
## Proportion of Variance    0.9917  0.0016  0.00144  0.00122  0.00098  0.00093
## Cumulative Proportion     0.9917  0.9933  0.99471  0.99593  0.99691  0.99784
##                             PC7      PC8      PC9      PC10
## Standard deviation     65.29428 59.90981 53.20803 2.637e-13
## Proportion of Variance  0.00086  0.00073  0.00057 0.000e+00
## Cumulative Proportion   0.99870  0.99943  1.00000 1.000e+00
```

Do our PCA plot of this RNA-Seq data.

```
plot(pca1$x[,1], pca1$x[,2], xlab="PC1", ylab="PC2")
```




```
plot(pca1$x[,1], pca1$x[,2], xlab="PC1", ylab="PC2")
text(pca1$x[,1], pca1$x[,2], colnames(rna.data))
```



Optional: Gene loadings

What measurements(genes) contribute the most to PC1 in either direction (+ or -)?

```
loading_scores <- pca1$rotation[,1]

# Find the top 10 measurements (genes) that contribute most to PC1 in either direction
gene_scores <- abs(loading_scores)
gene_scores_ranked <- sort(gene_scores, decreasing = TRUE)

# Show the name of the top 10 genes
top_10_genes <- names(gene_scores_ranked[1:10])
top_10_genes
```

```
##  [1] "gene98" "gene45" "gene10" "gene21" "gene48" "gene50" "gene18" "gene62"
##  [9] "gene3"  "gene60"
```