Class 7: Machine Learning

James Garza (PID: A16300772)

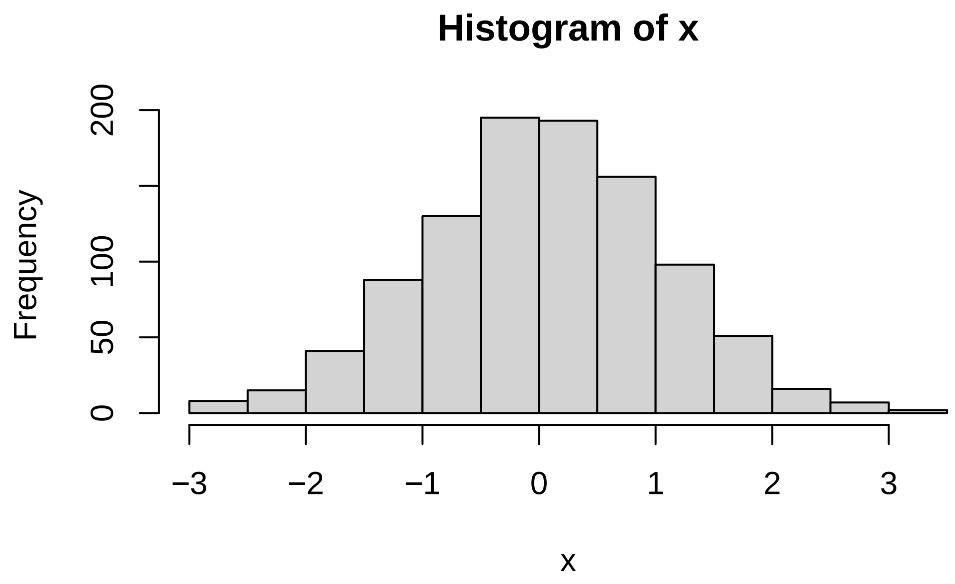
Clustering Methods

The broad goal here is to find groupings (clusters) in your input data.

Kmeans

First, let's make up some data to cluster.

```
x <- rnorm(1000)
hist(x)
```



Make a vector of lenght 60 with 30 points centered at -3 and 30 points centered at +3

```
tmp <- c(rnorm(30, mean = -3), rnorm(30, mean = 3))
tmp
```

```
 [1] -3.2497001 -2.9960754 -3.5405584 -3.5362546 -4.2808990 -4.1227896
 [7] -4.4939644 -3.5758856 -2.2662447 -4.7061522 -1.8127652 -4.7904719
[13] -2.0838015 -3.4638459 -0.7977986 -4.1084967 -3.4849455 -3.1003737
[19] -3.9655195 -3.2277651 -3.1205836 -1.7273039 -2.6697502 -1.8550035
[25] -1.4460594 -3.0891307 -2.1826058 -2.9852604 -3.5334903 -1.9907298
[31]  6.1463842  2.8843271  1.4224898  2.9262609  1.3951402  3.1124970
[37]  5.9501037  2.3472971  3.6844099  2.0533744  3.9850708  1.8918611
[43]  3.7117273  4.7068738  4.3311282  1.7697299  4.2033186  4.3216588
[49]  2.5407410  3.0515339  2.1919909  3.3063703  3.2505730  2.0918038
[55]  3.3082360  2.6550818  2.8831643  3.3027545  4.8344482  2.5007855
```

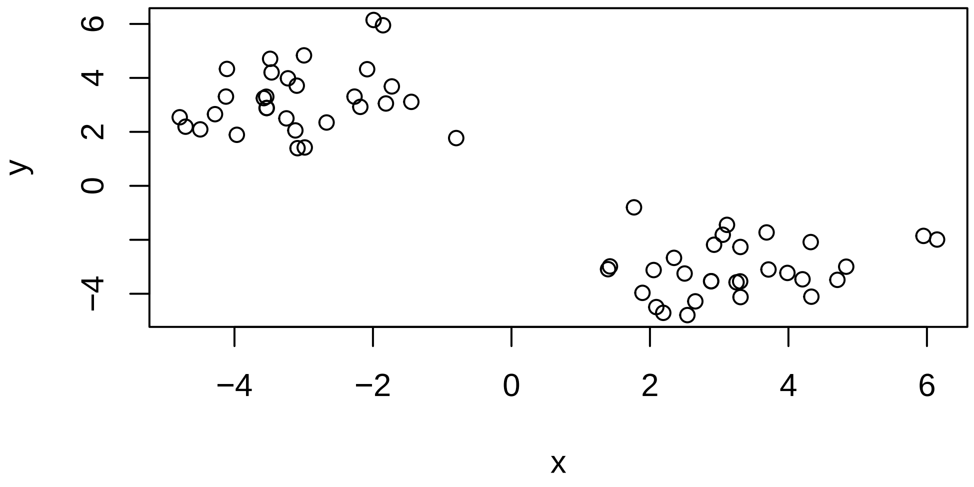
I will now make a wee x and y dataset with 2 groups of points.

```
x <- cbind(x=tmp, y=rev(tmp))
x
```

```
               x          y
 [1,] -3.2497001  2.5007855
 [2,] -2.9960754  4.8344482
 [3,] -3.5405584  3.3027545
 [4,] -3.5362546  2.8831643
 [5,] -4.2808990  2.6550818
 [6,] -4.1227896  3.3082360
 [7,] -4.4939644  2.0918038
 [8,] -3.5758856  3.2505730
 [9,] -2.2662447  3.3063703
[10,] -4.7061522  2.1919909
[11,] -1.8127652  3.0515339
[12,] -4.7904719  2.5407410
[13,] -2.0838015  4.3216588
[14,] -3.4638459  4.2033186
[15,] -0.7977986  1.7697299
[16,] -4.1084967  4.3311282
[17,] -3.4849455  4.7068738
[18,] -3.1003737  3.7117273
[19,] -3.9655195  1.8918611
```

```
[20,] -3.2277651  3.9850708
[21,] -3.1205836  2.0533744
[22,] -1.7273039  3.6844099
[23,] -2.6697502  2.3472971
[24,] -1.8550035  5.9501037
[25,] -1.4460594  3.1124970
[26,] -3.0891307  1.3951402
[27,] -2.1826058  2.9262609
[28,] -2.9852604  1.4224898
[29,] -3.5334903  2.8843271
[30,] -1.9907298  6.1463842
[31,]  6.1463842 -1.9907298
[32,]  2.8843271 -3.5334903
[33,]  1.4224898 -2.9852604
[34,]  2.9262609 -2.1826058
[35,]  1.3951402 -3.0891307
[36,]  3.1124970 -1.4460594
[37,]  5.9501037 -1.8550035
[38,]  2.3472971 -2.6697502
[39,]  3.6844099 -1.7273039
[40,]  2.0533744 -3.1205836
[41,]  3.9850708 -3.2277651
[42,]  1.8918611 -3.9655195
[43,]  3.7117273 -3.1003737
[44,]  4.7068738 -3.4849455
[45,]  4.3311282 -4.1084967
[46,]  1.7697299 -0.7977986
[47,]  4.2033186 -3.4638459
[48,]  4.3216588 -2.0838015
[49,]  2.5407410 -4.7904719
[50,]  3.0515339 -1.8127652
[51,]  2.1919909 -4.7061522
[52,]  3.3063703 -2.2662447
[53,]  3.2505730 -3.5758856
[54,]  2.0918038 -4.4939644
[55,]  3.3082360 -4.1227896
[56,]  2.6550818 -4.2808990
[57,]  2.8831643 -3.5362546
[58,]  3.3027545 -3.5405584
[59,]  4.8344482 -2.9960754
[60,]  2.5007855 -3.2497001
```

```
plot(x)
```



kmeans(x, centers, iter.max = 10, nstart = 1, algorithm = c("Hartigan-Wong", "Lloyd", "Forgy", "MacQueen"), trace = FALSE)

centers is how many groups/clusters there

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

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

Cluster means:
          x         y
1  3.225371 -3.073474
2 -3.073474  3.225371

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 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 1 1

Within cluster sum of squares by cluster:
```

```
[1] 71.66617 71.66617
 (between_SS / total_SS =  89.3 %)

Available components:

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

Q. Form your result object `k` how many points are in each cluster?

```
k$size
```

```
[1] 30 30
```

Q. What "component" of your result object details the cluster membership?

```
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 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?

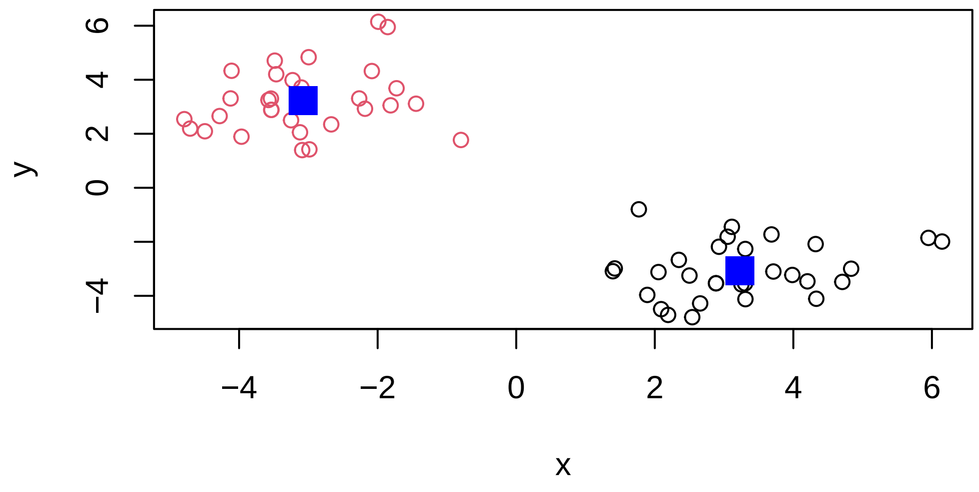
```
k$centers
```

```
          x         y
1  3.225371 -3.073474
2 -3.073474  3.225371
```

Q. Plot of our cluster results

You can color plots by number

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



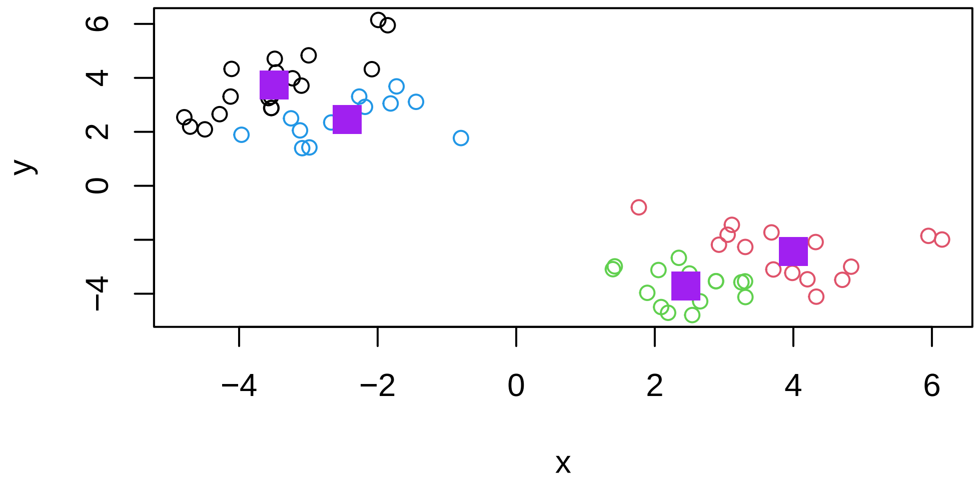
Cluster the data into 4 groups now

```
# kmeans
k_4 <- kmeans (x, centers = 4)
k_4
```

```
K-means clustering with 4 clusters of sizes 18, 15, 15, 12

Cluster means:
          x         y
1 -3.493972  3.738855
2  4.002768 -2.436254
3  2.447975 -3.710694
4 -2.442727  2.455146

Clustering vector:
 [1] 4 1 1 1 1 1 1 1 4 1 4 1 1 1 4 1 1 1 4 1 4 4 4 1 4 4 4 4 1 1 2 3 3 2 3 2 2 3
[39] 2 3 2 3 2 2 2 2 2 2 3 2 3 2 3 3 3 3 3 3 2 3

Within cluster sum of squares by cluster:
[1] 36.54683 29.98171 11.37263 15.29756
 (between_SS / total_SS =  93.0 %)
```

```
Available components:

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

```
# Plot of Results with 4 Centers
plot(x, col=k_4$cluster)
points(k_4$centers, col="purple", pch=15, cex=2)
```



A big limitation of kmeans is that it does what you ask even if you ask for silly clusters.

hclust() i.e. Hierarchical Clustering

hclust(d, method = "complete", members = NULL)

S3 method for class 'hclust' plot(x, labels = NULL, hang = 0.1, check = TRUE, axes = TRUE, frame.plot = FALSE, ann = TRUE, main = "Cluster Dendrogram", sub = NULL, xlab = NULL, ylab = "Height", ...)

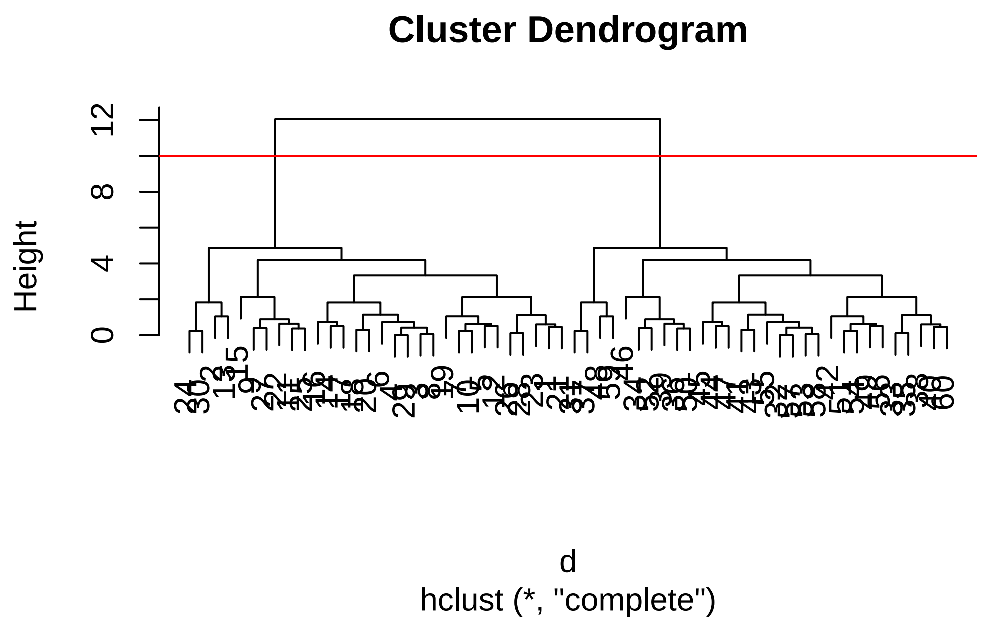
The main base R function for Hierarchical Clustering is `hclust()`. Unlike `kmeans()` you can bot just pass your data as input. You first need to calculate a distance matrix.

```r
# distance matrix
d <- dist(x)
# then hierarchical clustering
hc <- hclust(d)
hc
```

```
Call:
hclust(d = d)

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

Use `plot()` to view these results

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



The dendrogram naturally diveraged around the two centers where the first center was 1-30 and the second centers was for number 31-60 Height indicates how far apart the labels are, lower is closer together

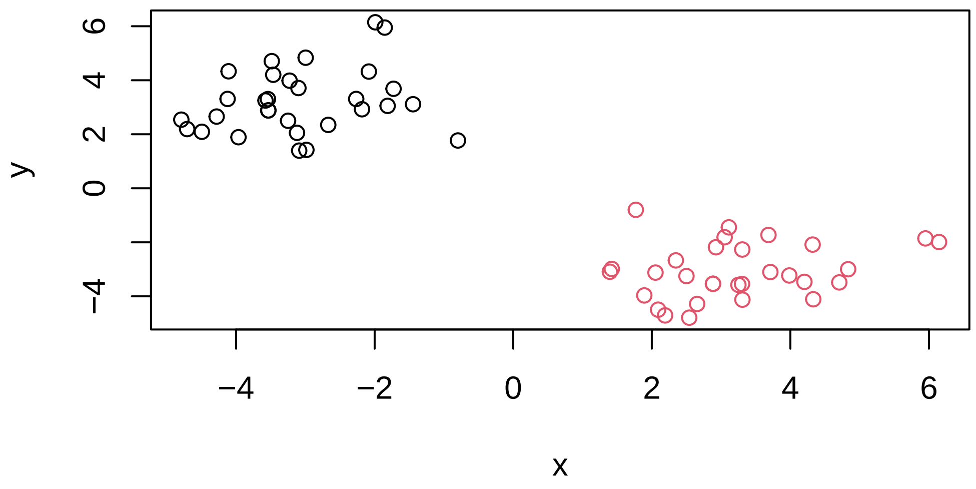
To make the "cut" and get out cluster membership vector we can use the `cutree()` function.

```
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 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
```

Make a plot of our data colored by hcluster results

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



Principal Component Analysis (PCA)

Here we will do Principal Component Analysis (PCA) on some food data from the UK.

#Lab 7

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

```
# View(x)
```

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

```
## Complete the following code to find out how many rows and columns are in x?
row(x)
```

```
      [,1] [,2] [,3] [,4]
 [1,]    1    1    1    1
 [2,]    2    2    2    2
 [3,]    3    3    3    3
 [4,]    4    4    4    4
 [5,]    5    5    5    5
 [6,]    6    6    6    6
 [7,]    7    7    7    7
 [8,]    8    8    8    8
 [9,]    9    9    9    9
[10,]   10   10   10   10
[11,]   11   11   11   11
[12,]   12   12   12   12
[13,]   13   13   13   13
[14,]   14   14   14   14
[15,]   15   15   15   15
[16,]   16   16   16   16
[17,]   17   17   17   17
```

```
col(x)
```

```
      [,1] [,2] [,3] [,4]
 [1,]    1    2    3    4
 [2,]    1    2    3    4
 [3,]    1    2    3    4
 [4,]    1    2    3    4
 [5,]    1    2    3    4
 [6,]    1    2    3    4
 [7,]    1    2    3    4
 [8,]    1    2    3    4
 [9,]    1    2    3    4
[10,]    1    2    3    4
```

```
[11,]    1    2    3    4
[12,]    1    2    3    4
[13,]    1    2    3    4
[14,]    1    2    3    4
[15,]    1    2    3    4
[16,]    1    2    3    4
[17,]    1    2    3    4
```

```r
## Preview the first 6 rows
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
```

```r
# 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
```

```r
dim(x)
```

```
[1] 17  4
```

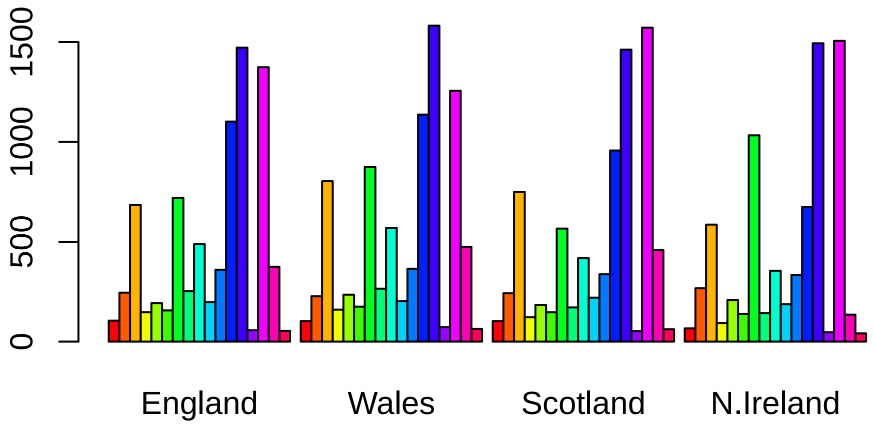
```r
x <- read.csv(url, row.names=1)
head(x)
```

```
               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 option because it does not self override itself each time it is run as the first code we used was self-destructive and eventually resulted in an error if we kept on running it. Again the second approach is more robust because it can be run multiple times without canabalizing the data.

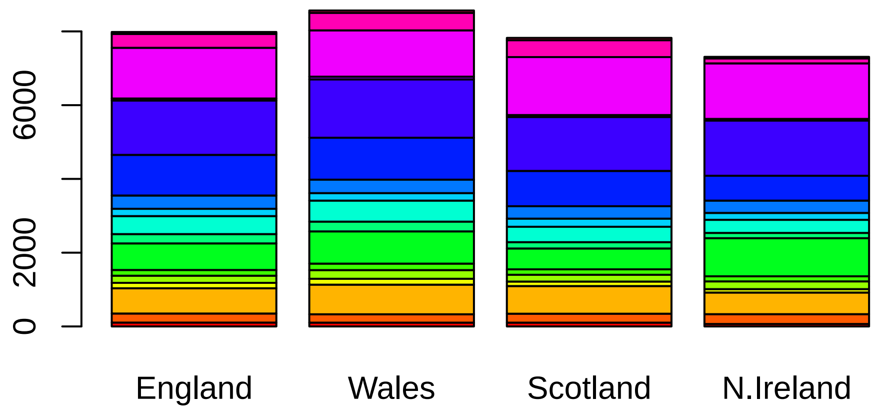
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



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

The optional argument that needs to be removed is "beside=T".

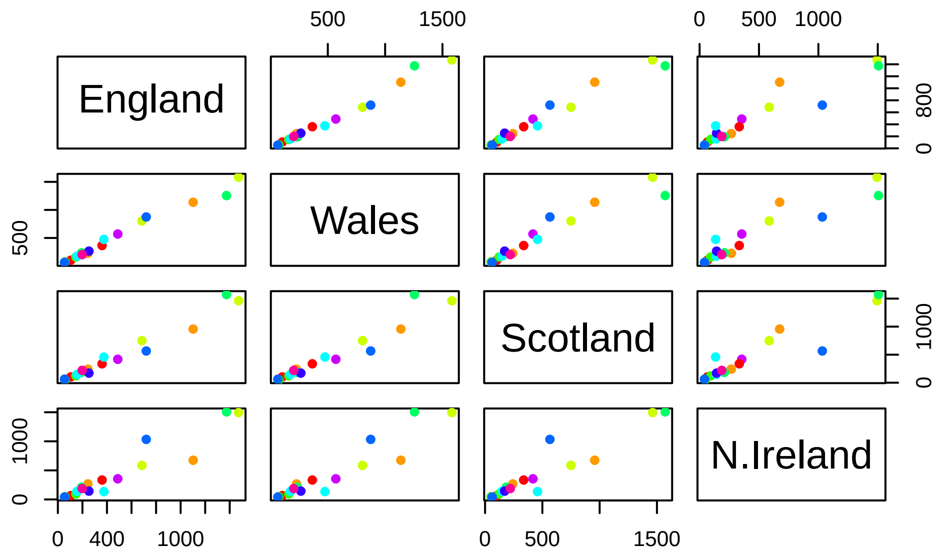
```r
barplot(as.matrix(x), col=rainbow(nrow(x)))
```



> 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 code below finds the pairs between the values that are matching within the countries against each other. The y-axis is all a specific country in that row so each graph represents each pair of countries plotted against each other. If a given point is on a diagonal there would be no difference in that category for the two countries being compared.

```r
pairs(x, col=rainbow(10), pch=16)
```



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

In terms of the data set above the main differences in each of the graphs with N. Ireland is that the points are not as diagonal which indicates there is a greater difference in the consumption of foods that are not on the diagonal line. Additionally the blue and orange points are very different to other countries.

PCA to the Rescue

The main "base" R function for PCA is called `prcomp()`. Here we need to take the transpose of our input as we want the countries in the rows and food as the columns.

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

```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 3.176e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

> Q. How much variance is captured in 2 PCs?

96.5%, PC1 captures 67.44% and PC2 captures 29.05%

To make our main “PC score plot” (a.k.a. “PC1 vs PC2 plot”, or “PC plot” or “ordination plot”).

```
attributes(pca)
```

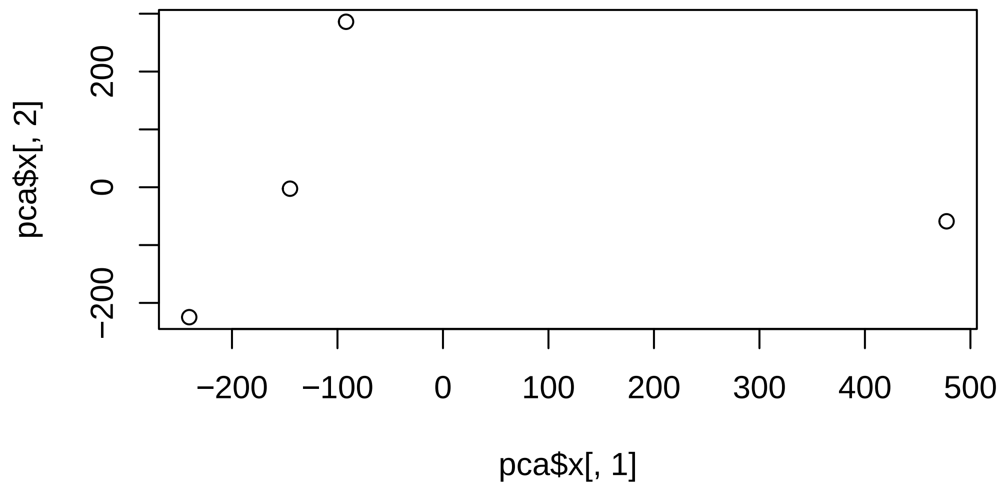
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"       

$class
[1] "prcomp"
```

We are after the `pca$x` result component to make our main PCA plot.

```
pca$x
```

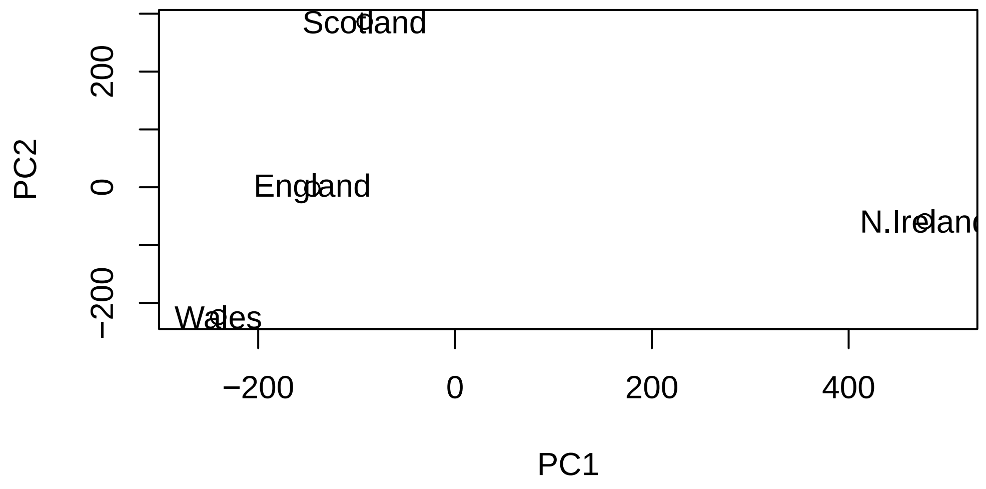
```
                 PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -4.894696e-14
Wales     -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland   -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland  477.39164  -58.901862  -4.877895  2.321303e-13
```

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



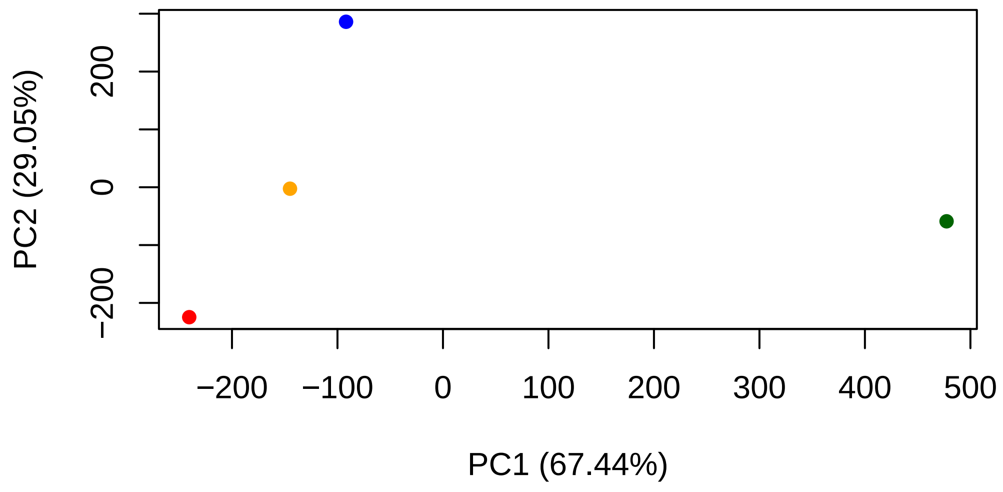
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```r
# 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.

```
mycols <- c("orange", "red", "blue", "darkgreen")
plot(pca$x[,1], pca$x[,2], col=mycols, pch=16, xlab="PC1 (67.44%)", ylab="PC2 (29.05%)")
```



Another important result from PCA is how the original variables (in this case the the foods) contribute to the PCs.

This contained in teh `pca$rotation` object - folks often call this the "loadings" or "contributions" to the PCs

```
pca$rotation
```

```
                            PC1          PC2         PC3          PC4
Cheese              -0.056955380  0.016012850  0.02394295 -0.694538519
Carcass_meat         0.047927628  0.013915823  0.06367111  0.489884628
Other_meat          -0.258916658 -0.015331138 -0.55384854  0.279023718
Fish                -0.084414983 -0.050754947  0.03906481 -0.008483145
Fats_and_oils       -0.005193623 -0.095388656 -0.12522257  0.076097502
Sugars              -0.037620983 -0.043021699 -0.03605745  0.034101334
Fresh_potatoes       0.401402060 -0.715017078 -0.20668248 -0.090972715
Fresh_Veg           -0.151849942 -0.144900268  0.21382237 -0.039901917
Other_Veg           -0.243593729 -0.225450923 -0.05332841  0.016719075
Processed_potatoes  -0.026886233  0.042850761 -0.07364902  0.030125166
Processed_Veg       -0.036488269 -0.045451802  0.05289191 -0.013969507
Fresh_fruit         -0.632640898 -0.177740743  0.40012865  0.184072217
Cereals             -0.047702858 -0.212599678 -0.35884921  0.191926714
```

```
Beverages           -0.026187756 -0.030560542 -0.04135860  0.004831876
Soft_drinks          0.232244140  0.555124311 -0.16942648  0.103508492
Alcoholic_drinks    -0.463968168  0.113536523 -0.49858320 -0.316290619
Confectionery       -0.029650201  0.005949921 -0.05232164  0.001847469
```

```
max(pca$rotation[1,])
```

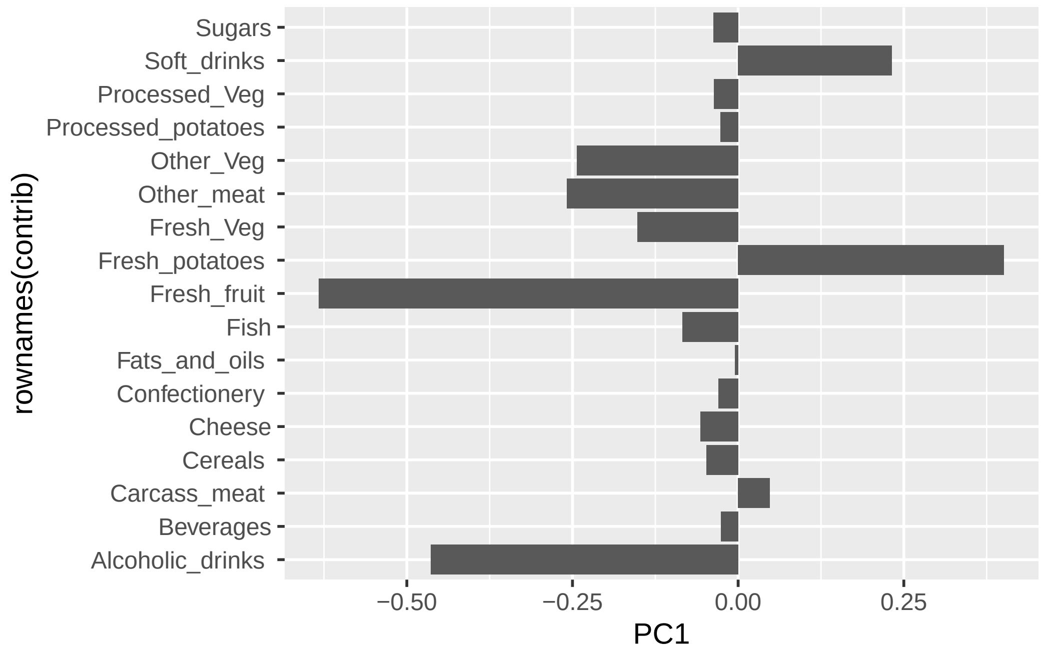
```
[1] 0.02394295
```

```
min(pca$rotation[1,])
```

```
[1] -0.6945385
```

We can make a plot along PC1

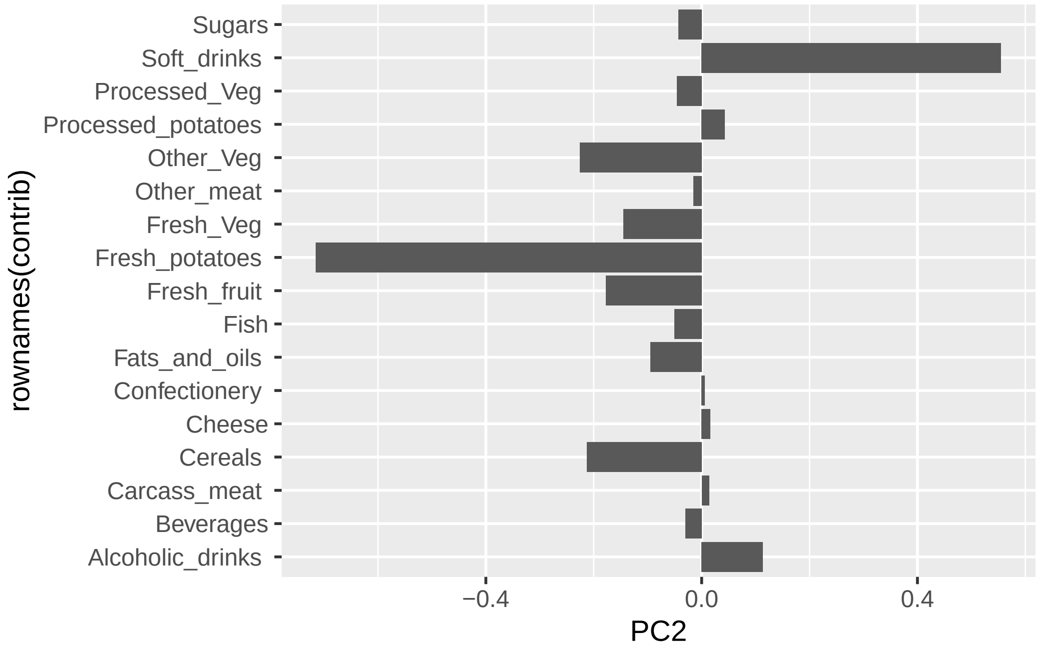
```
library(ggplot2)

contrib <- as.data.frame((pca$rotation))

ggplot(contrib) +
  aes(PC1, rownames(contrib)) + geom_col()
```



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

The 2 food groups featured prominently are fresh potatoes and soft drinks. PC2 mainly tells us that second variance is driven by the differences in fresh potatoes and soft drinks.

```
ggplot(contrib) +
  aes(PC2, rownames(contrib)) + geom_col()
```


Sugars
Soft_drinks
Processed_Veg
Processed_potatoes
Other_Veg
Other_meat
Fresh_Veg
Fresh_potatoes
Fresh_fruit
Fish
Fats_and_oils
Confectionery
Cheese
Cereals
Carcass_meat
Beverages
Alcoholic_drinks
rownames(contrib)
PC2