Class 7: Machine Learning 1

Hailey Wheeler (A13312713)

Clustering

We will start with k-means clustering (one of the most popular of all methods)

```
rnorm(10)
```

```
 [1]  0.68714969 -0.08274258 -0.58560820 -0.19247768  0.79771872  0.28673551
 [7] -0.74229988  0.52163456  0.15098643 -0.97339975
```

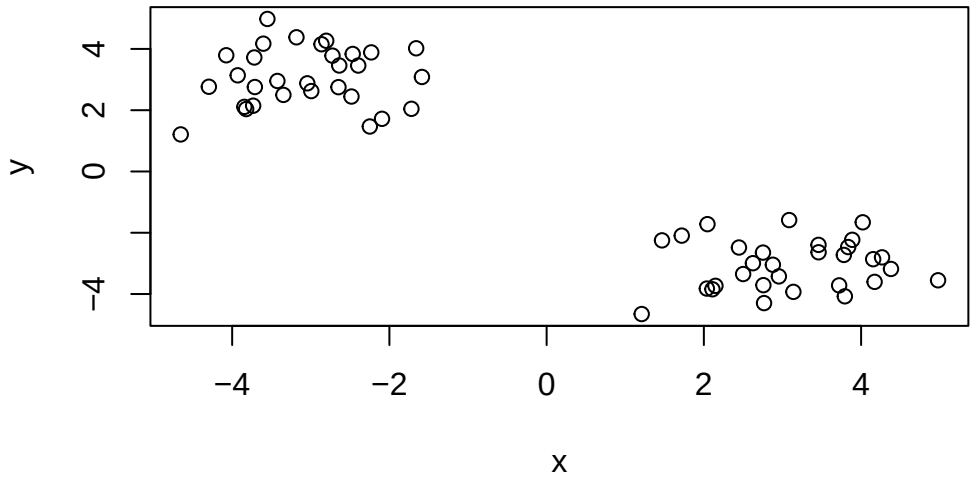
```
tmp <- c(rnorm(30,3), rnorm(30,-3))
x <- cbind(x=tmp, y=rev(tmp))
x
```

```
              x         y
 [1,]  3.720726 -3.718353
 [2,]  2.446384 -2.482332
 [3,]  1.209203 -4.654161
 [4,]  2.756881 -3.711131
 [5,]  2.954061 -3.424459
 [6,]  3.138584 -3.929406
 [7,]  2.624093 -2.993458
 [8,]  3.459265 -2.637717
 [9,]  3.780839 -2.724028
[10,]  4.379901 -3.181930
[11,]  2.045585 -1.719657
[12,]  3.084916 -1.586082
[13,]  2.147122 -3.732844
[14,]  3.793141 -4.074983
[15,]  2.108886 -3.844831
```

```
[16,]  3.458230 -2.396071
[17,]  4.020951 -1.659330
[18,]  3.887556 -2.230635
[19,]  4.153298 -2.863368
[20,]  2.752040 -2.647668
[21,]  1.718635 -2.094043
[22,]  4.171704 -3.603595
[23,]  4.979066 -3.551665
[24,]  4.266472 -2.803412
[25,]  3.835503 -2.466817
[26,]  2.765382 -4.296450
[27,]  2.037746 -3.822097
[28,]  2.499507 -3.348299
[29,]  1.467783 -2.248936
[30,]  2.877689 -3.044023
[31,] -3.044023  2.877689
[32,] -2.248936  1.467783
[33,] -3.348299  2.499507
[34,] -3.822097  2.037746
[35,] -4.296450  2.765382
[36,] -2.466817  3.835503
[37,] -2.803412  4.266472
[38,] -3.551665  4.979066
[39,] -3.603595  4.171704
[40,] -2.094043  1.718635
[41,] -2.647668  2.752040
[42,] -2.863368  4.153298
[43,] -2.230635  3.887556
[44,] -1.659330  4.020951
[45,] -2.396071  3.458230
[46,] -3.844831  2.108886
[47,] -4.074983  3.793141
[48,] -3.732844  2.147122
[49,] -1.586082  3.084916
[50,] -1.719657  2.045585
[51,] -3.181930  4.379901
[52,] -2.724028  3.780839
[53,] -2.637717  3.459265
[54,] -2.993458  2.624093
[55,] -3.929406  3.138584
[56,] -3.424459  2.954061
[57,] -3.711131  2.756881
[58,] -4.654161  1.209203
```

```
[59,] -2.482332  2.446384
[60,] -3.718353  3.720726
```

```
plot(x)
```



The main function in R for K-mean clustering is called 'kmeans()'

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

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

Cluster means:
          x         y
1  3.084705 -3.049726
2 -3.049726  3.084705

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 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
```

```
Within cluster sum of squares by cluster:
[1] 45.12639 45.12639
 (between_SS / total_SS =  92.6 %)

Available components:

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

Lists "available components"

Question 1: How many points are in each cluster?

```
k$size
```

```
[1] 30 30
```

Question 2: The clustering result i.e. membership vector?

```
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 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
```

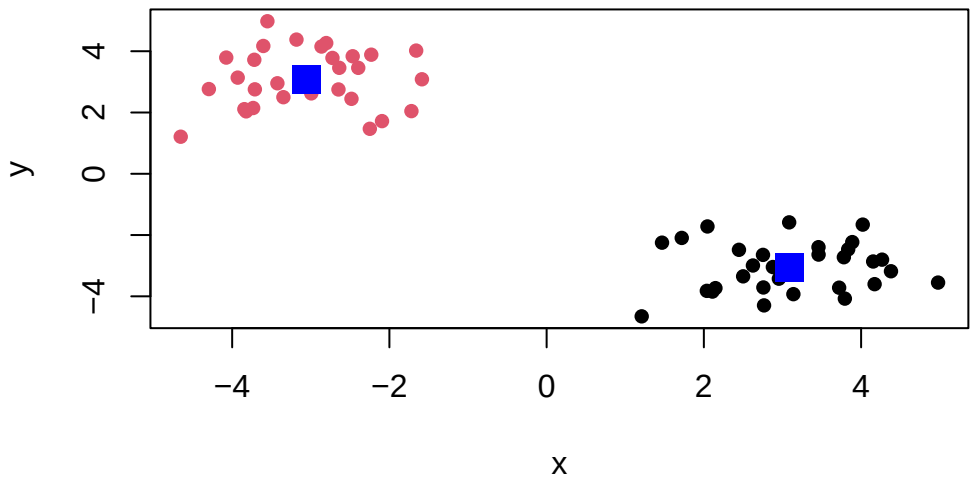
Question 3:

```
k$centers
```

```
          x         y
1  3.084705 -3.049726
2 -3.049726  3.084705
```

Question 4: Make a plot of our data colored by clustering results with optionally the cluster centers shown

```
plot(x, col = k$cluster, pch = 16)
points(k$centers, col="blue", pch = 15, cex = 2)
```



pch makes the plot easier to see cex alters size dynamics

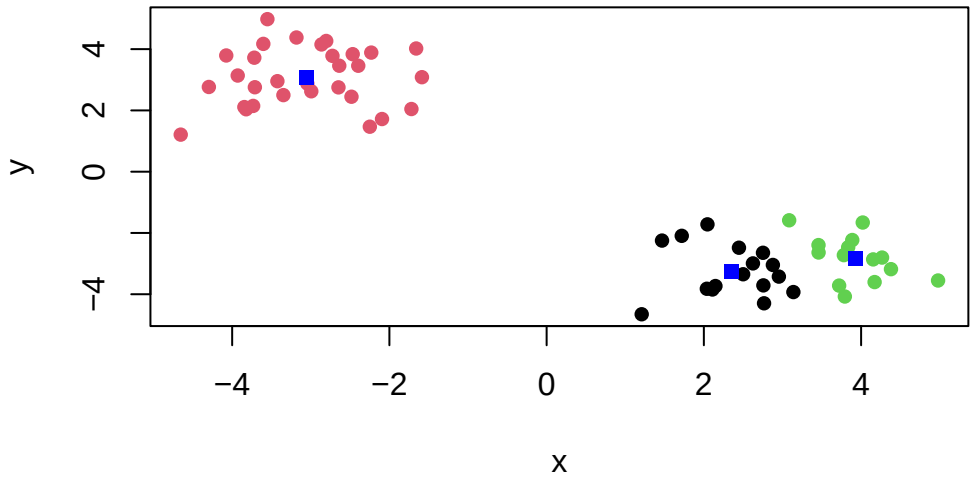
Question 5: Run kmeans again but cluster into 3 groups and plot the results like we did above

```
k3 <- kmeans(x, centers=3, nstart=20)
k3
```

```
K-means clustering with 3 clusters of sizes 16, 30, 14

Cluster means:
          x         y
1  2.346849 -3.249612
2 -3.049726  3.084705
3  3.927969 -2.821285

Clustering vector:
 [1] 3 1 1 1 1 1 1 3 3 3 1 3 1 3 1 3 3 3 3 1 1 3 3 3 3 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

Within cluster sum of squares by cluster:
[1] 15.07535 45.12639 10.01495
```

```
 (between_SS / total_SS =  94.2 %)

Available components:

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

```
plot(x, col = k3$cluster, pch = 16)
points(k3$centers, col="blue", pch = 15, cex = 1)
```



Hierarchical Clustering

Hierarchical clustering has an advantage inthat it can reveal the structure in your data rather than imposing a structure a sk-means will

The main function in 'base' R is called 'hclust()'

It requires a distance matrix as input, not the raw data itself

```
x
```

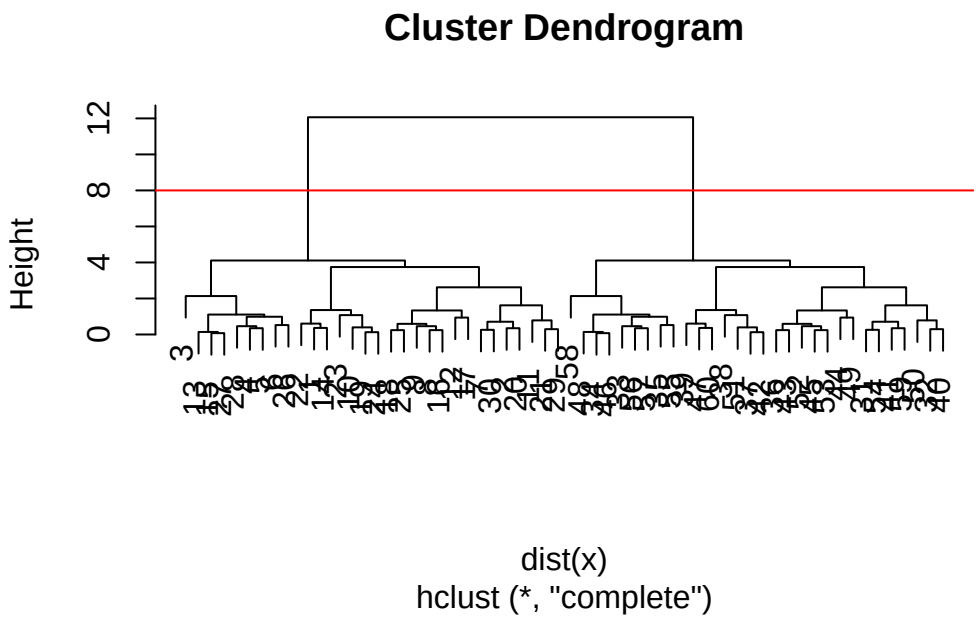
```
              x         y
 [1,]  3.720726 -3.718353
 [2,]  2.446384 -2.482332
 [3,]  1.209203 -4.654161
 [4,]  2.756881 -3.711131
 [5,]  2.954061 -3.424459
 [6,]  3.138584 -3.929406
 [7,]  2.624093 -2.993458
 [8,]  3.459265 -2.637717
 [9,]  3.780839 -2.724028
[10,]  4.379901 -3.181930
[11,]  2.045585 -1.719657
[12,]  3.084916 -1.586082
[13,]  2.147122 -3.732844
[14,]  3.793141 -4.074983
[15,]  2.108886 -3.844831
[16,]  3.458230 -2.396071
[17,]  4.020951 -1.659330
[18,]  3.887556 -2.230635
[19,]  4.153298 -2.863368
[20,]  2.752040 -2.647668
[21,]  1.718635 -2.094043
[22,]  4.171704 -3.603595
[23,]  4.979066 -3.551665
[24,]  4.266472 -2.803412
[25,]  3.835503 -2.466817
[26,]  2.765382 -4.296450
[27,]  2.037746 -3.822097
[28,]  2.499507 -3.348299
[29,]  1.467783 -2.248936
[30,]  2.877689 -3.044023
[31,] -3.044023  2.877689
[32,] -2.248936  1.467783
[33,] -3.348299  2.499507
[34,] -3.822097  2.037746
[35,] -4.296450  2.765382
[36,] -2.466817  3.835503
[37,] -2.803412  4.266472
[38,] -3.551665  4.979066
[39,] -3.603595  4.171704
[40,] -2.094043  1.718635
[41,] -2.647668  2.752040
[42,] -2.863368  4.153298
```

```
[43,] -2.230635  3.887556
[44,] -1.659330  4.020951
[45,] -2.396071  3.458230
[46,] -3.844831  2.108886
[47,] -4.074983  3.793141
[48,] -3.732844  2.147122
[49,] -1.586082  3.084916
[50,] -1.719657  2.045585
[51,] -3.181930  4.379901
[52,] -2.724028  3.780839
[53,] -2.637717  3.459265
[54,] -2.993458  2.624093
[55,] -3.929406  3.138584
[56,] -3.424459  2.954061
[57,] -3.711131  2.756881
[58,] -4.654161  1.209203
[59,] -2.482332  2.446384
[60,] -3.718353  3.720726
```

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

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

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

```r
plot(hc)
abline(h=8, col="red")
```



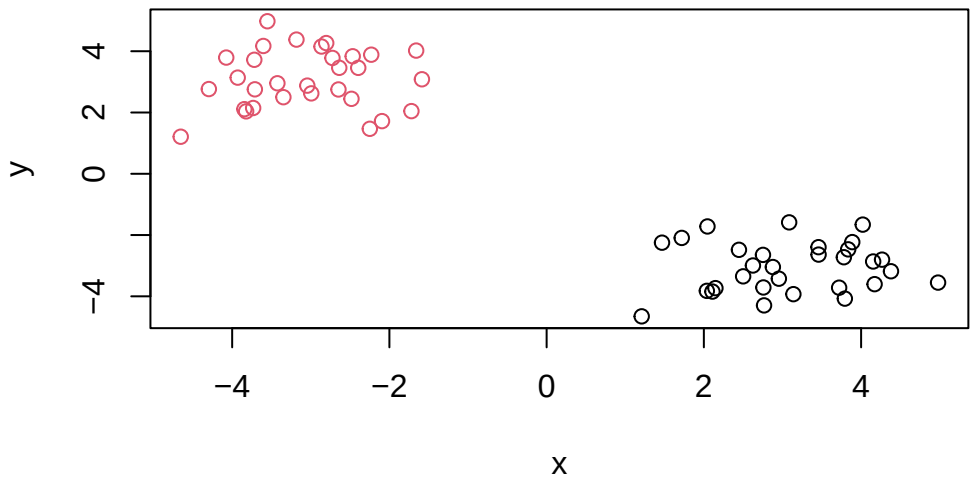
The function to get our clusters/groups from a hclust object is called 'cutree()'

```
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 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
```

Question: Plot our hclust results in terms of our data colored by cluster membership

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



kmeans(x,centers=2, nstart=20) hclust(dist(x))

In class Assignment

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

Q1: How many rows and columns are in your new data frame (x)? # Use dim(x) to identify # of rows, columns

```
ncol(x)
```

```
[1] 4
```

```
nrow(x)
```

```
[1] 17
```

```
dim(x)
```

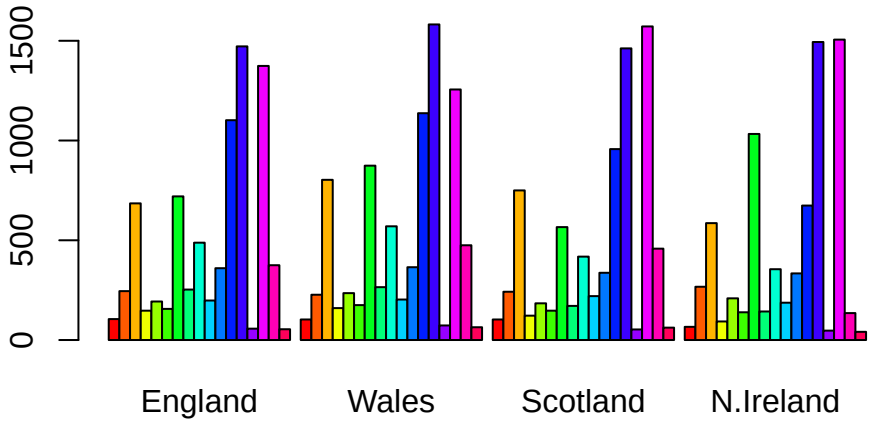
```
[1] 17  4
```

```
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
```

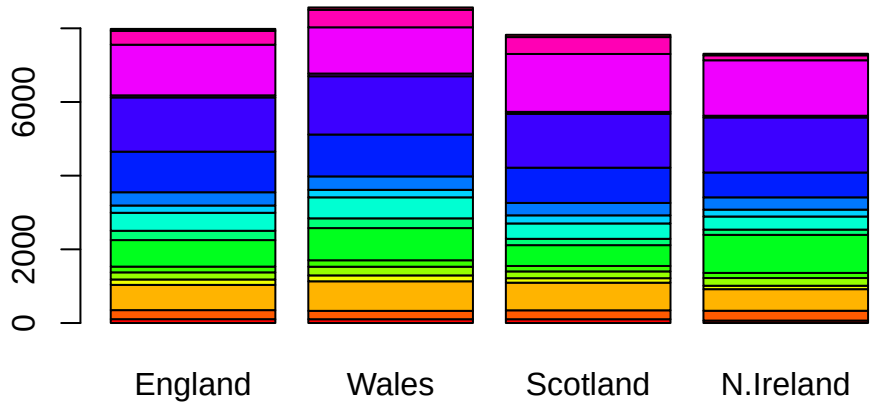
Q2: Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances? # I prefer setting the row.names = 1, when creating the dataframe. It is more robust and efficient compared to establishing row names as x[,1] and re-defining x data base.

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

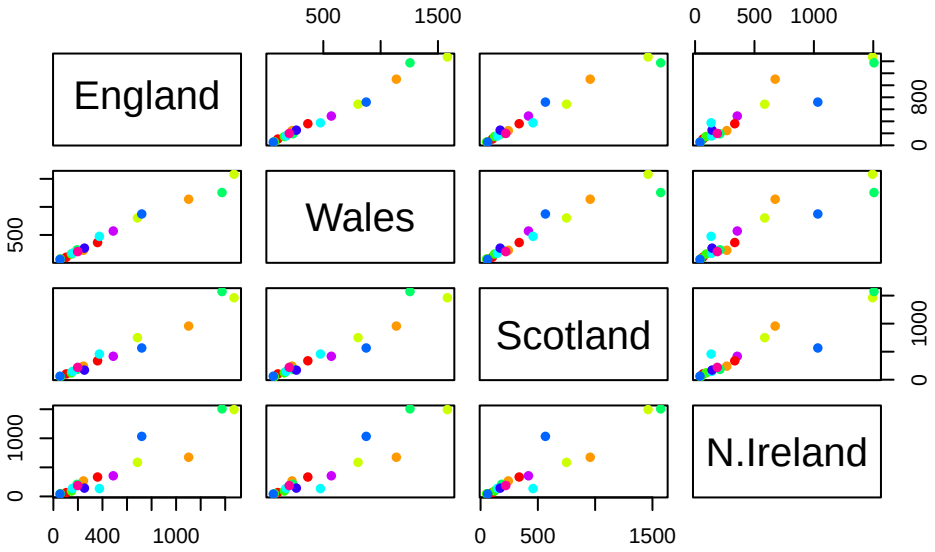


Q3: Changing what optional argument in the above barplot() function results in the following plot? # Changing beside = FALSE changes the "besides" comparison to the plot below

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




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



The more points that fit along the diagonal line, the more similar.

Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set? The blue dot in the Ireland data set is an outlier against the dot from UK.

PCA to the rescue

Help me make sense of this data The main function for PCA in base R is called 'prcomp()'

It wants the transpose (with the 't()'function) of our food data for analysis

```
t(x)
```

```
           Cheese Carcass_meat  Other_meat  Fish Fats_and_oils  Sugars
England       105           245         685  147            193    156
Wales         103           227         803  160            235    175
Scotland      103           242         750  122            184    147
N.Ireland      66           267         586   93            209    139
           Fresh_potatoes  Fresh_Veg  Other_Veg  Processed_potatoes
England               720        253        488                 198
Wales                 874        265        570                 203
Scotland              566        171        418                 220
N.Ireland            1033        143        355                 187
           Processed_Veg  Fresh_fruit  Cereals  Beverages Soft_drinks
England              360         1102     1472         57         1374
Wales                365         1137     1582         73         1256
Scotland             337          957     1462         53         1572
N.Ireland            334          674     1494         47         1506
           Alcoholic_drinks  Confectionery
England                 375             54
Wales                   475             64
Scotland                458             62
N.Ireland               135             41
```

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

One of the main results that folks look for is called the "score plot" aka PC plot, PC1 vs PC2 plot

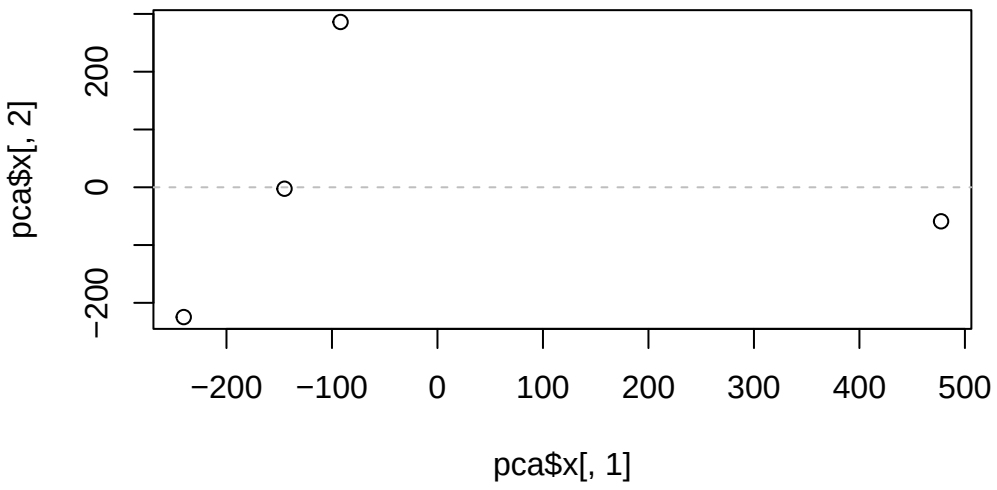
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

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.

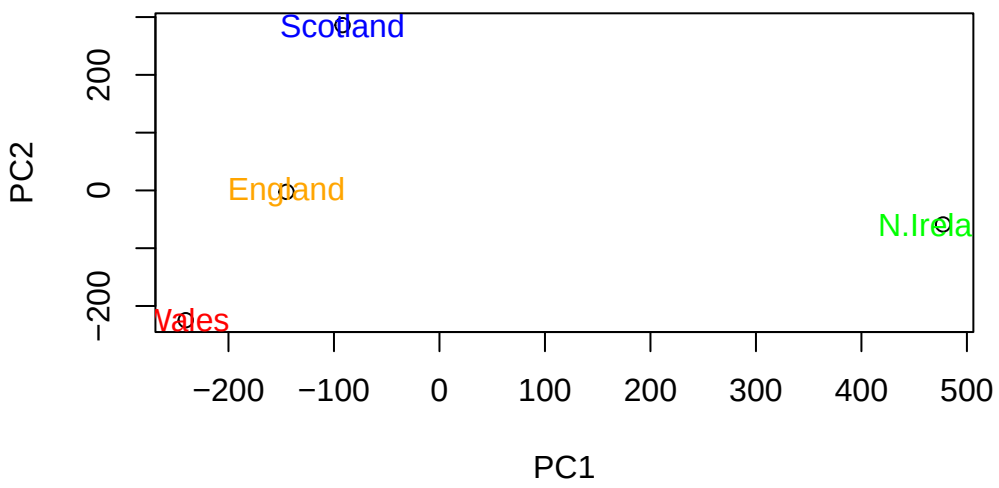
```
pca$x
```

```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -9.152022e-15
Wales     -240.52915 -224.646925 -56.475555  5.560040e-13
Scotland   -91.86934  286.081786 -44.415495 -6.638419e-13
N.Ireland  477.39164  -58.901862  -4.877895  1.329771e-13
```

```
plot(pca$x[,1], pca$x[,2])
abline(h=0, col="grey", lty=2)
```



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



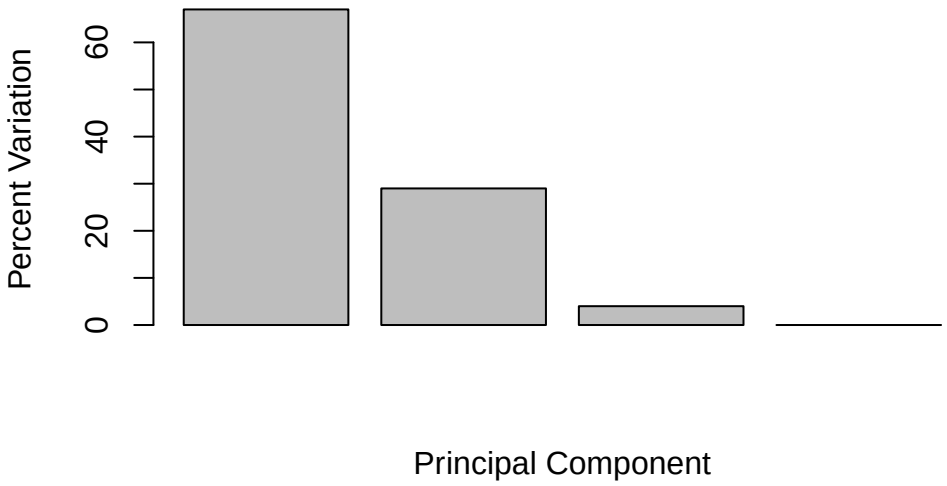

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

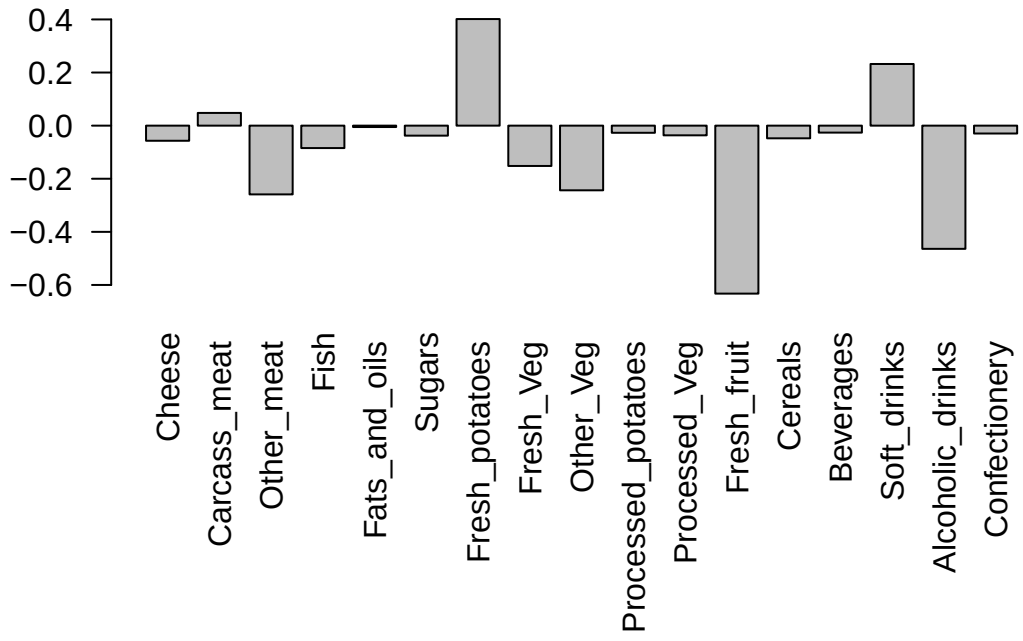
```
z <- summary(pca)
z$importance
```

```
                             PC1       PC2      PC3           PC4
Standard deviation     324.15019 212.74780 73.87622 2.921348e-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")
```

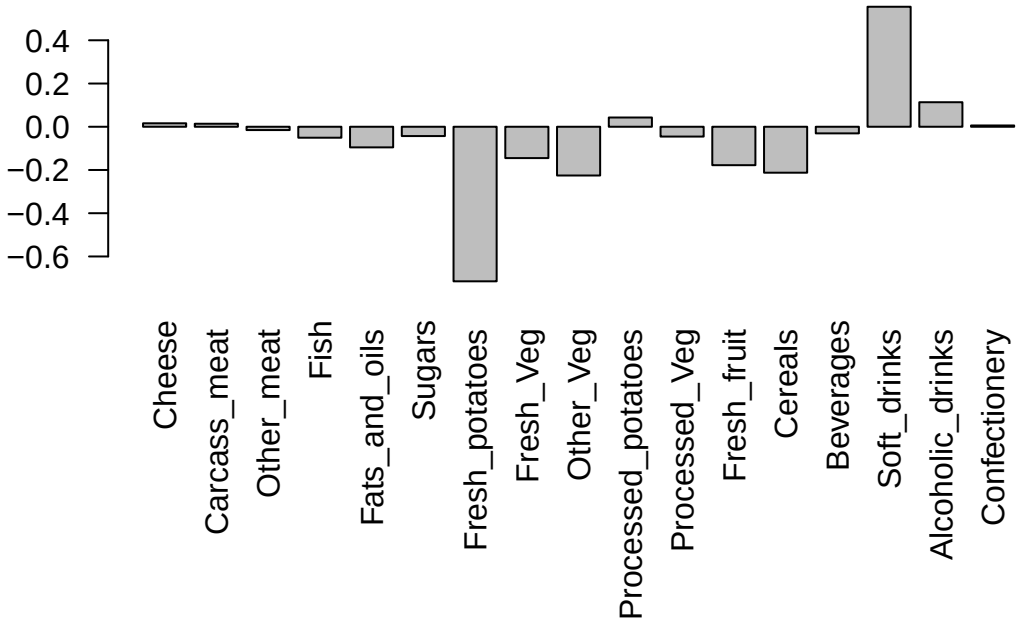



```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```

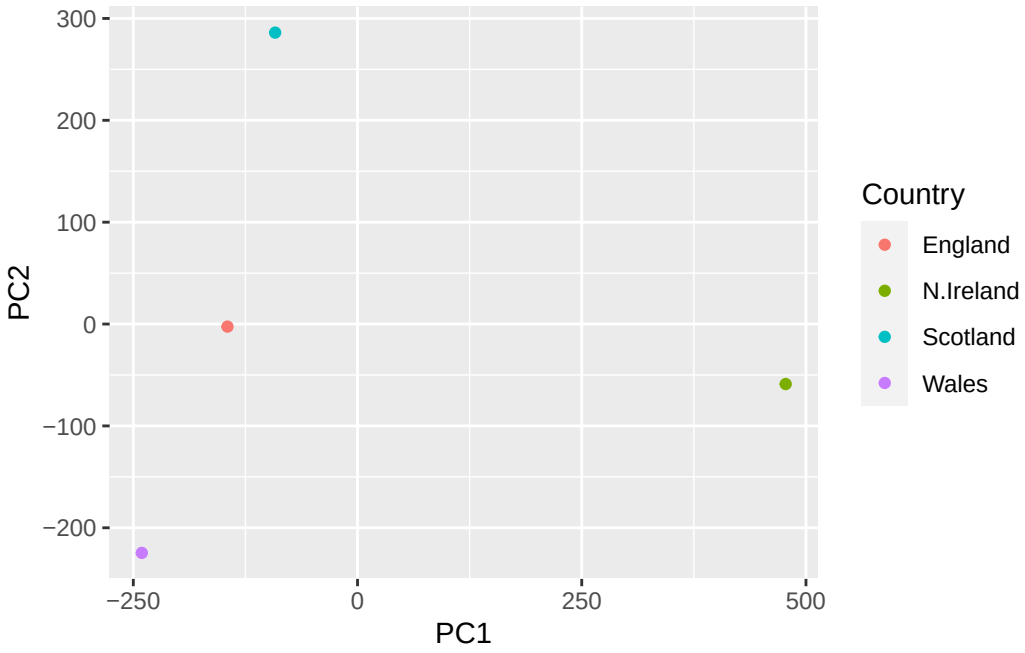


Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominently and what does PC2 maninly tell us about? The potatoes and soft_drinks. Wales eats more potatoes and drinks fewer soft drinks, as indicated by the positive and negative values. If its negative, it correlates with the negative number on PC2. In PC2, Wales has a negative number, meaning that it eats the most potatoes compared to the other countries.

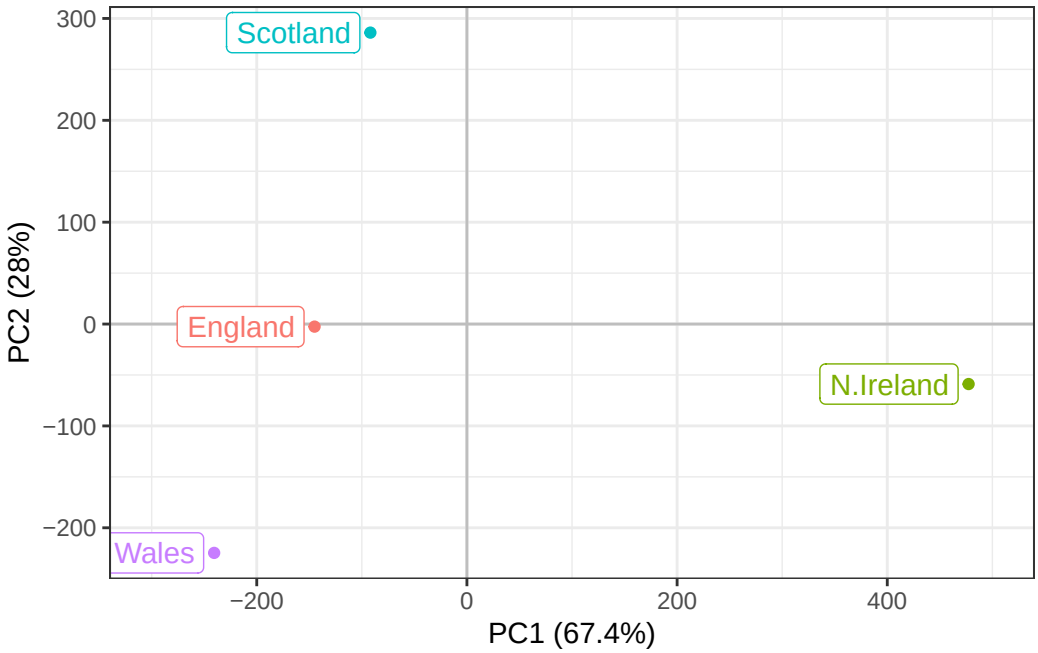
```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```



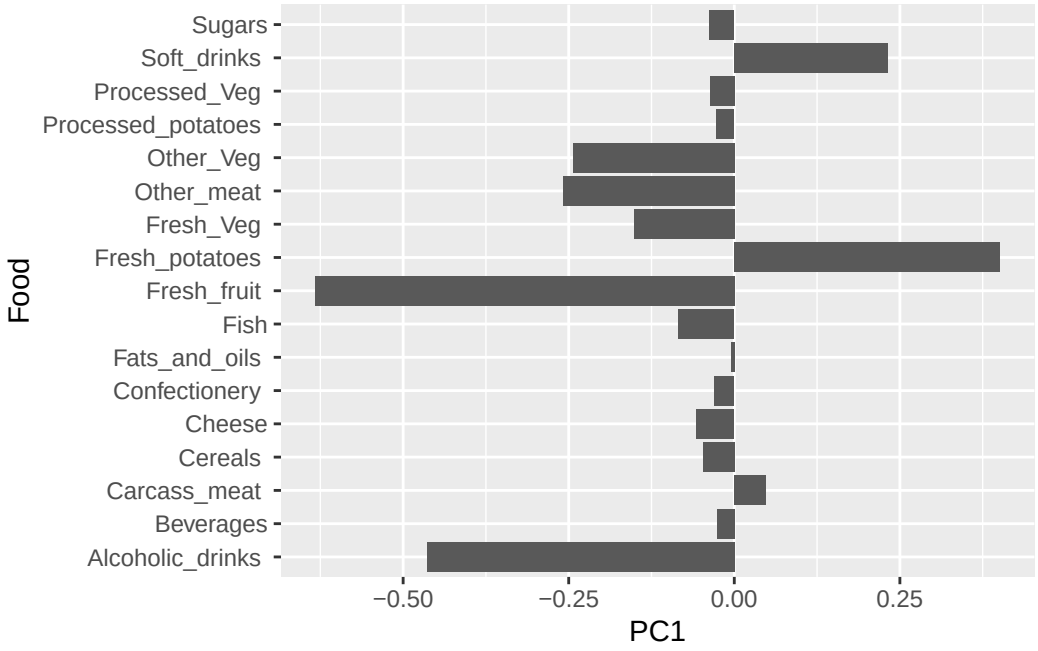

```
library(ggplot2)
df <- as.data.frame(pca$x)
df_lab <- tibble::rownames_to_column(df, "Country")

# Our first basic plot
ggplot(df_lab) +
  aes(PC1, PC2, col=Country) +
  geom_point()
```



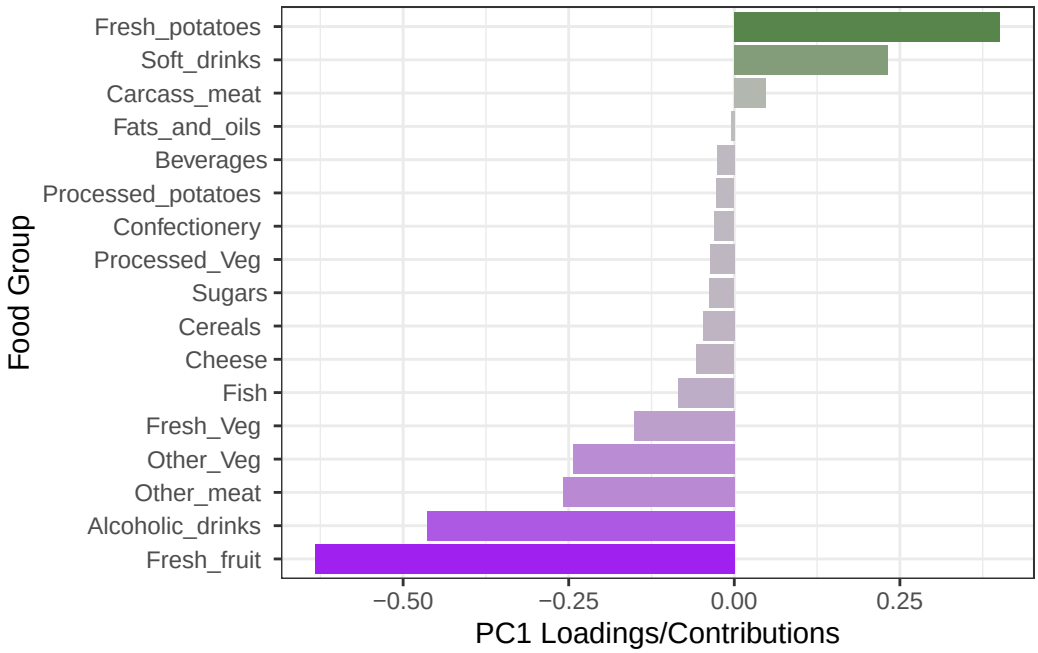

```
ggplot(df_lab) +
  aes(PC1, PC2, col=Country, label=Country) +
  geom_hline(yintercept = 0, col="gray") +
  geom_vline(xintercept = 0, col="gray") +
  geom_point(show.legend = FALSE) +
  geom_label(hjust=1, nudge_x = -10, show.legend = FALSE) +
  expand_limits(x = c(-300,500)) +
  xlab("PC1 (67.4%)") +
  ylab("PC2 (28%)") +
  theme_bw()
```



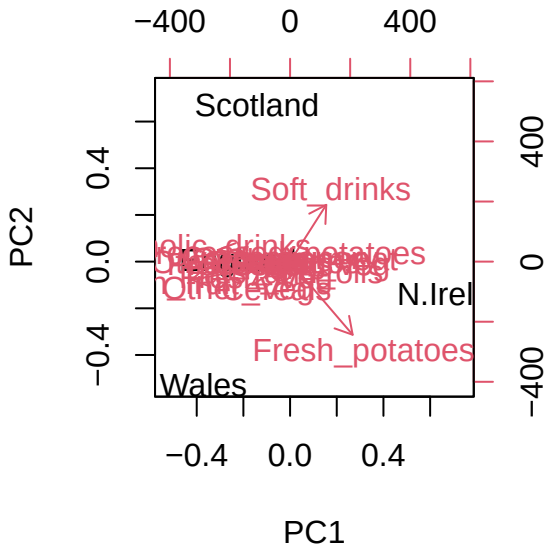

```
ld <- as.data.frame(pca$rotation)
ld_lab <- tibble::rownames_to_column(ld, "Food")

ggplot(ld_lab) +
  aes(PC1, Food) +
  geom_col()
```




```
ggplot(ld_lab) +
  aes(PC1, reorder(Food, PC1), bg=PC1) +
  geom_col() +
  xlab("PC1 Loadings/Contributions") +
  ylab("Food Group") +
  scale_fill_gradient2(low="purple", mid="gray", high="darkgreen", guide=NULL) +
  theme_bw()
```


Fresh_potatoes
Soft_drinks
Carcass_meat
Fats_and_oils
Beverages
Processed_potatoes
Confectionery
Processed_Veg
Sugars
Cereals
Cheese
Fish
Fresh_Veg
Other_Veg
Other_meat
Alcoholic_drinks
Fresh_fruit
Food Group
−0.50
−0.25
0.00
0.25
PC1 Loadings/Contributions

```
biplot(pca)
```


−400
0
400
Scotland
Soft_drinks
N.Irel
Fresh_potatoes
Wales
0.4
0.0
−0.4
PC2
400
0
−400
−0.4
0.0
0.4
PC1

RNA SEQ

```
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 10 samples (indicated by column) and 100 genes.

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

```
[1] 100  10
```