class07

Jaewon Kim

#First up kmeans()

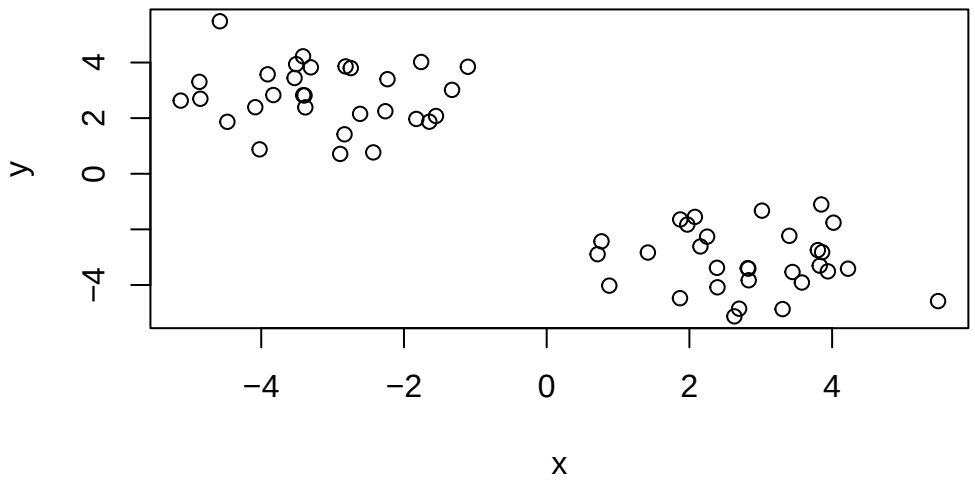
Demo od using kmeans() function in base R. First make up some data with a known structure

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

```
                x          y
 [1,] -2.4299630  0.7680160
 [2,] -4.5794768  5.4845289
 [3,] -3.4116203  2.8258086
 [4,] -3.4141044  4.2227216
 [5,] -4.0833213  2.3930204
 [6,] -1.8265841  1.9708359
 [7,] -3.3053700  3.8264845
 [8,] -3.3933442  2.8172204
 [9,] -5.1280109  2.6304323
[10,] -1.7590851  4.0191598
[11,] -3.5328789  3.4443407
[12,] -1.3263065  3.0168912
[13,] -1.5514610  2.0770164
[14,] -2.7454028  3.7998552
[15,] -3.8305330  2.8308448
[16,] -3.9103711  3.5761752
[17,] -1.1042096  3.8476628
[18,] -4.0228576  0.8774159
[19,] -2.8329438  1.4182653
[20,] -3.5102971  3.9401940
[21,] -3.3819886  2.3864066
[22,] -4.8665270  3.3058434
[23,] -2.2313263  3.4000682
```

```
[24,] -2.8183539  3.8596566
[25,] -2.8935800  0.7132048
[26,] -4.8528359  2.6965927
[27,] -2.2609426  2.2482116
[28,] -2.6144288  2.1541224
[29,] -1.6439931  1.8716139
[30,] -4.4744788  1.8681600
[31,]  1.8681600 -4.4744788
[32,]  1.8716139 -1.6439931
[33,]  2.1541224 -2.6144288
[34,]  2.2482116 -2.2609426
[35,]  2.6965927 -4.8528359
[36,]  0.7132048 -2.8935800
[37,]  3.8596566 -2.8183539
[38,]  3.4000682 -2.2313263
[39,]  3.3058434 -4.8665270
[40,]  2.3864066 -3.3819886
[41,]  3.9401940 -3.5102971
[42,]  1.4182653 -2.8329438
[43,]  0.8774159 -4.0228576
[44,]  3.8476628 -1.1042096
[45,]  3.5761752 -3.9103711
[46,]  2.8308448 -3.8305330
[47,]  3.7998552 -2.7454028
[48,]  2.0770164 -1.5514610
[49,]  3.0168912 -1.3263065
[50,]  3.4443407 -3.5328789
[51,]  4.0191598 -1.7590851
[52,]  2.6304323 -5.1280109
[53,]  2.8172204 -3.3933442
[54,]  3.8264845 -3.3053700
[55,]  1.9708359 -1.8265841
[56,]  2.3930204 -4.0833213
[57,]  4.2227216 -3.4141044
[58,]  2.8258086 -3.4116203
[59,]  5.4845289 -4.5794768
[60,]  0.7680160 -2.4299630
```

```
plot(x)
```



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

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

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

Cluster means:
          x         y
1  2.809692 -3.124553
2 -3.124553  2.809692

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] 71.76132 71.76132
 (between_SS / total_SS =  88.0 %)

Available components:
```

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

Q. How many points in each cluster

```
k$size
```

```
[1] 30 30
```

Q. How do we get to the cluster membership/assignment

```
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
```

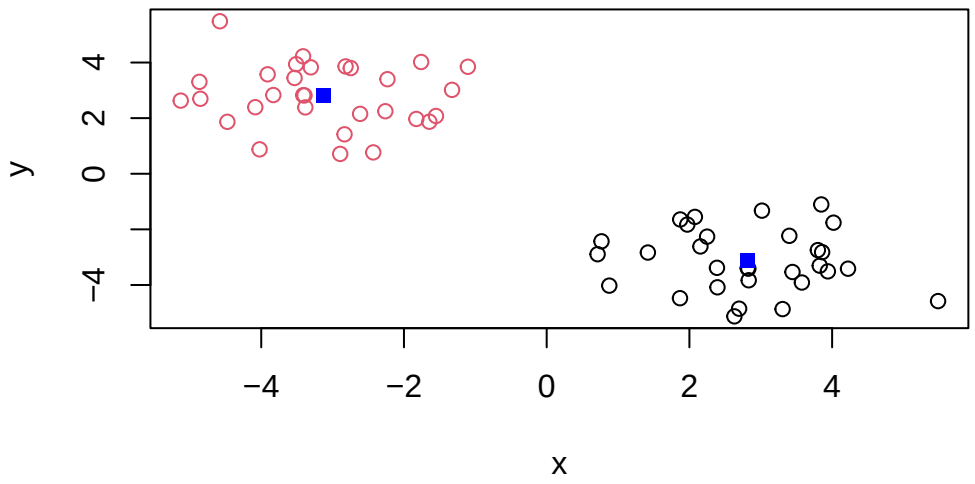
Q. What about cluster centers?

```
k$centers
```

```
          x         y
1  2.809692 -3.124553
2 -3.124553  2.809692
```

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

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



##Now for hclust()

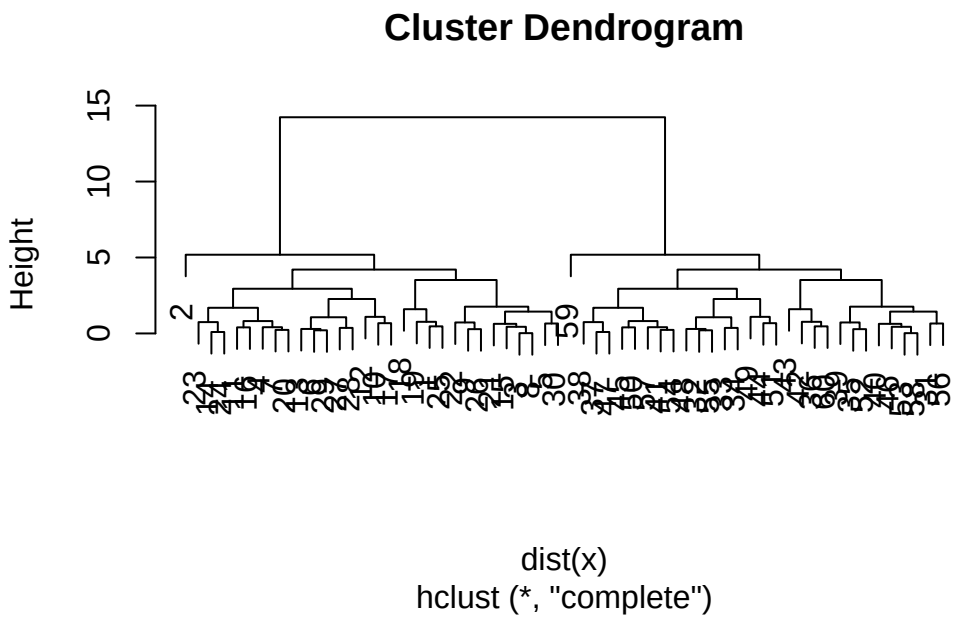
We will cluster the same data 'x' with the 'hclust()' In this case, 'hclust()' requires a distance matrix as input.

```
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)
```



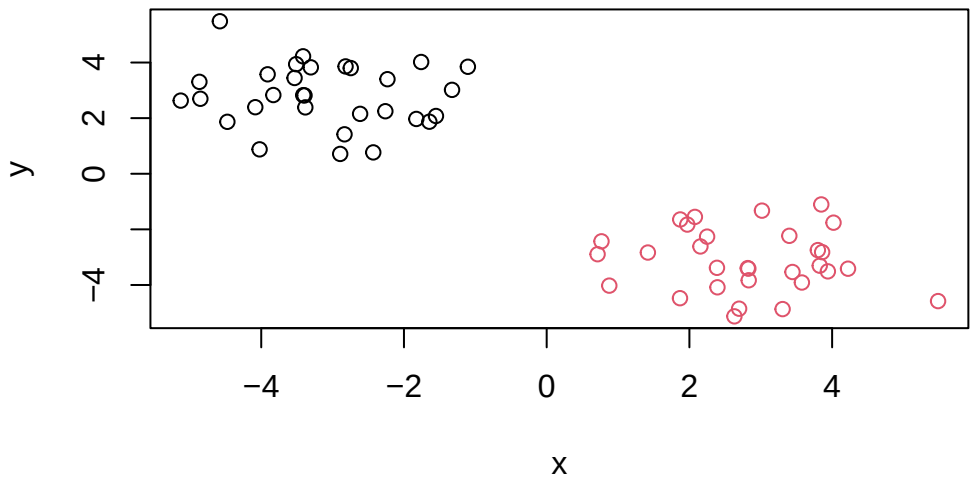
To get our cluster membership vector, we need to "cut" the tree with the 'cutree()'

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

```
 [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 2
```

Now plot our hclust() results

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



Q. Import UK_foods.csv

```
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?

```
dim(x)
```

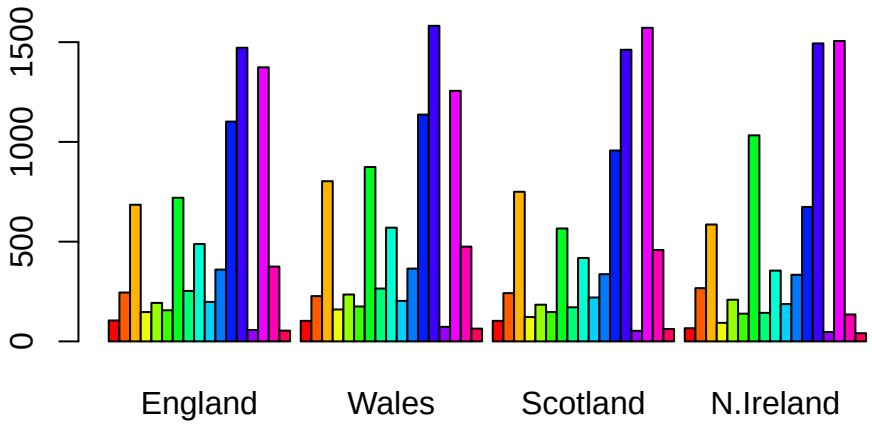
```
[1] 17  4
```

```
#There are 4 columns and 17 rows.
```

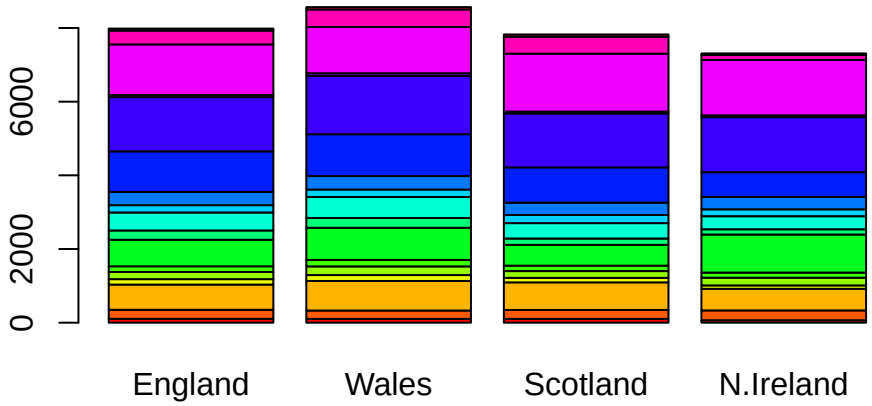
Q2. Which approach to solving the 'row-names problem' mentioned above do you prefer and why? Is one approach more robust than another under certain circumstances?

```
#row.names() is preferred. Using index requires variable assignment. However, R can store
```

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

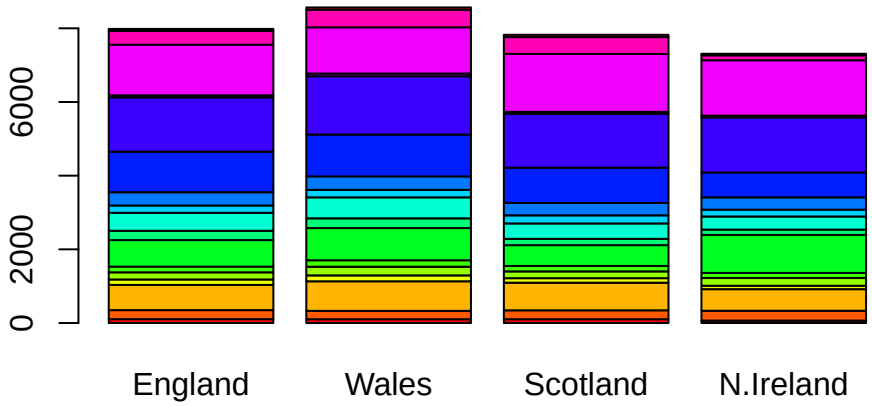


```
cols <- rainbow(nrow(x))
barplot(as.matrix(x), col = cols)
```



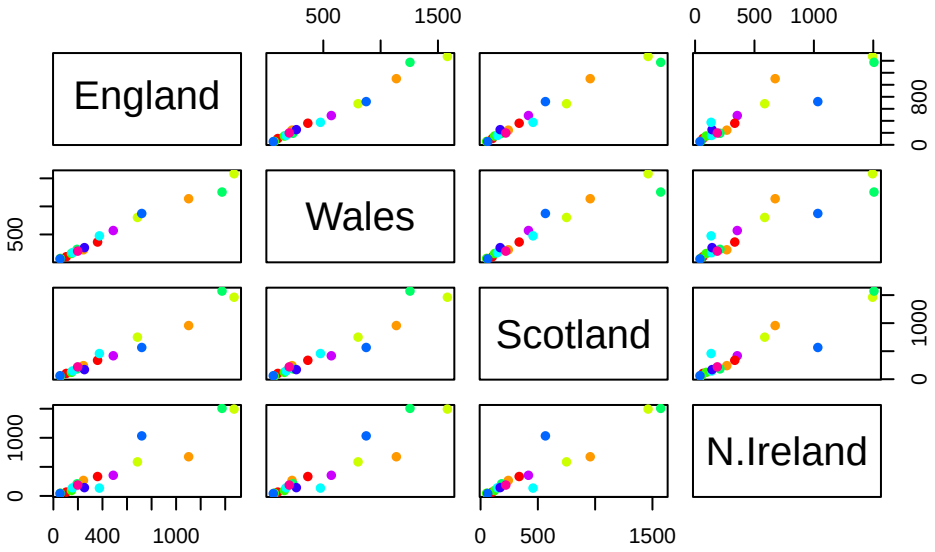
Q3: Changing what optional argument in the above barplot() function results in the following plot? (stacked one)

```
#Remove besides = True (or change true to false) makes all data stacked within same countr

barplot(as.matrix(x), beside=F, 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?

```
#Y-axis in each row corresponds to consumption in country in same row. X-axis in each colu

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?

```
#Blue dot where N.Ireland is on y-axis is higher (low where N.Ireland is on x-axis) than d
```

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

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

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

$class
[1] "prcomp"
```

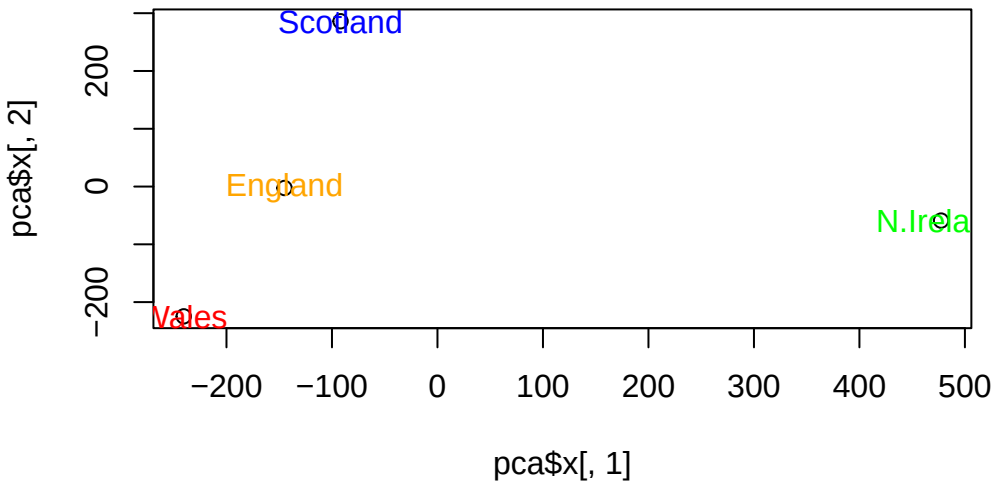
```
pca
```

```
Standard deviations (1, .., p=4):
```

```
[1] 3.241502e+02 2.127478e+02 7.387622e+01 3.175833e-14

Rotation (n x k) = (17 x 4):
                           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
```

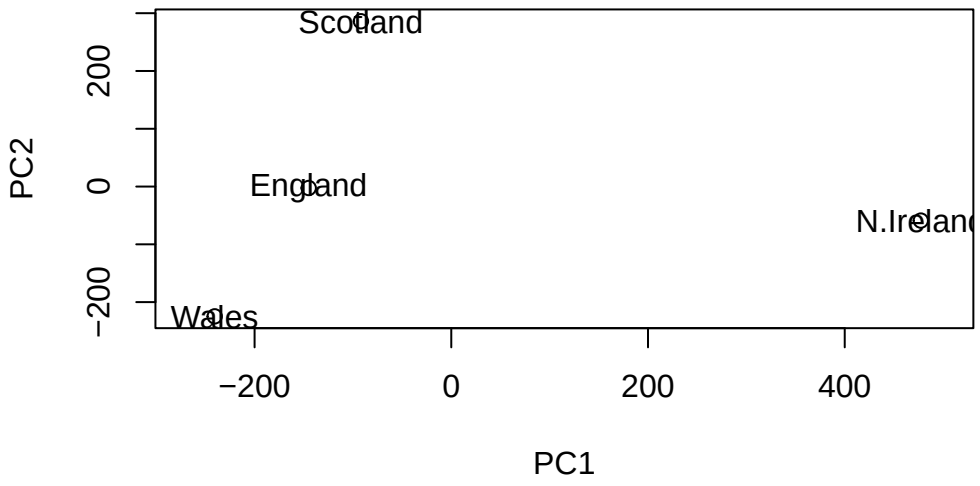
To make our new PCA plot (a.k.a PCA scores) we access 'pca$x'

```
country_cols <- c("orange", "red", "blue", "green")
plot(pca$x[,1], pca$x[,2])
text(pca$x[,1], pca$x[,2], colnames(x), col = country_cols)
```



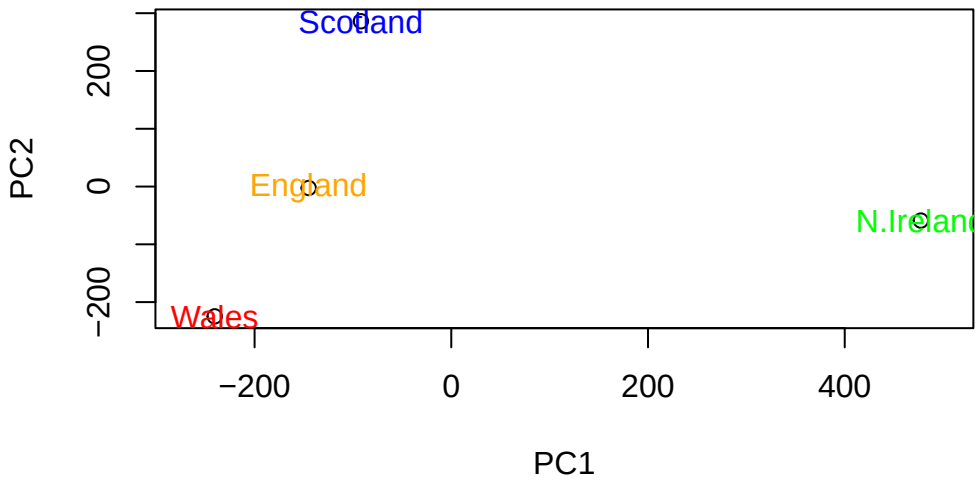
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
plot(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)
```



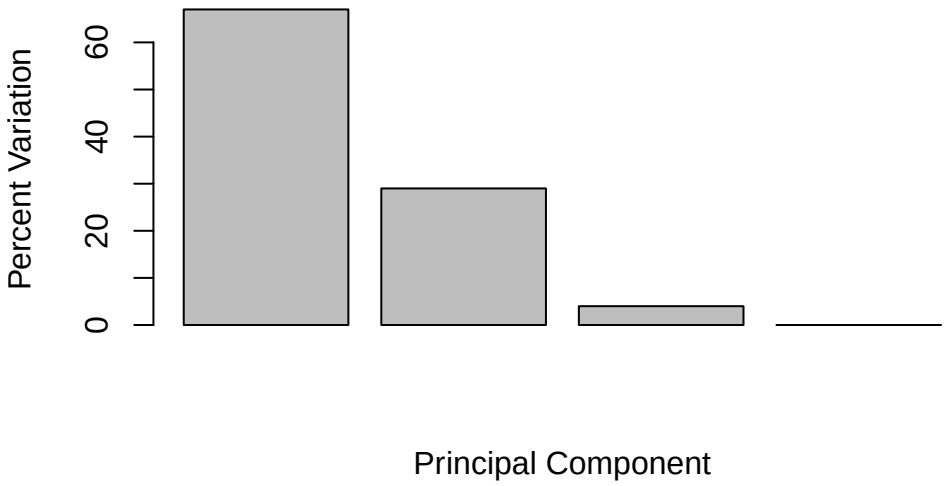

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

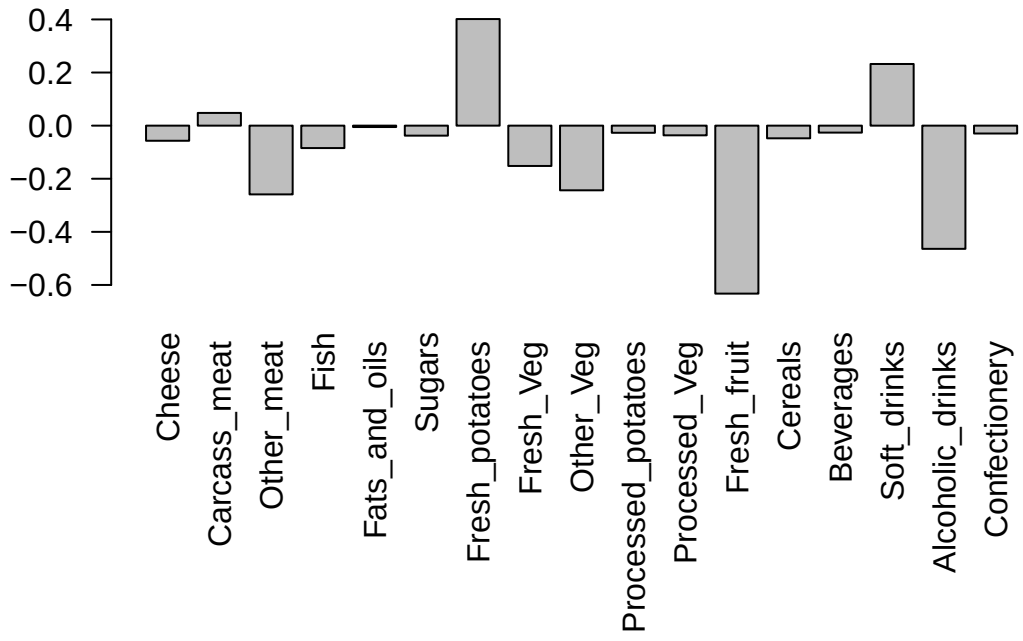
```
z <- summary(pca)
z$importance
```

```
                             PC1       PC2      PC3           PC4
Standard deviation     324.15019 212.74780 73.87622 3.175833e-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")
```

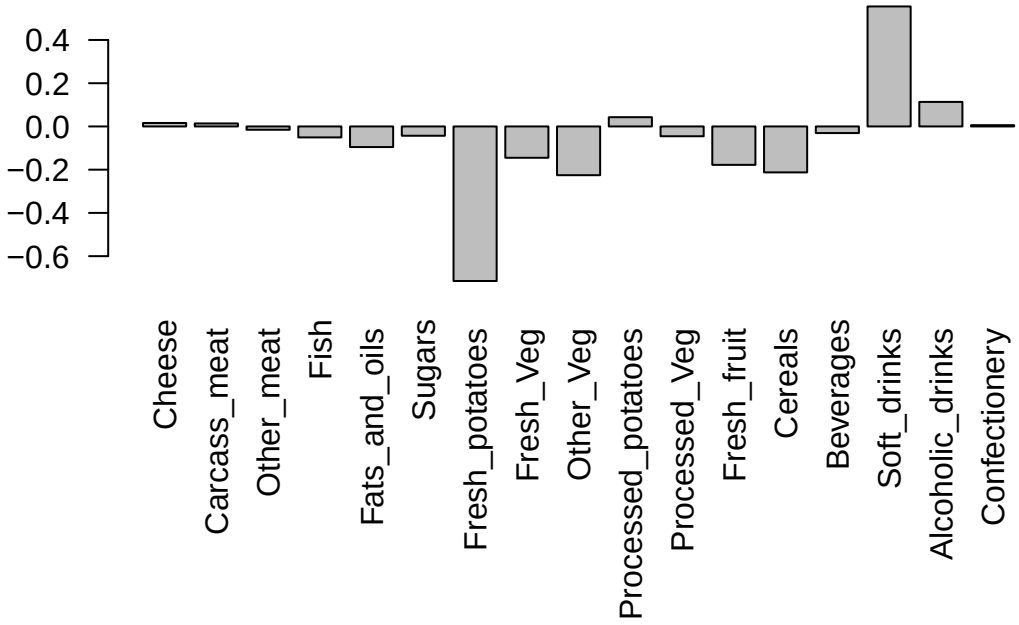



```
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```



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

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




```
# PC2 indicates axis that captures variance second.
# Fresh_potatoes pushes country with high potato consumption down
# and ones with high soft drink consumption pusehd up on plot.
# This is shown in PCA plot, where England/N.Ireland has similar
# food consumption pattern, thus levels similar on PC2.
```

```
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?

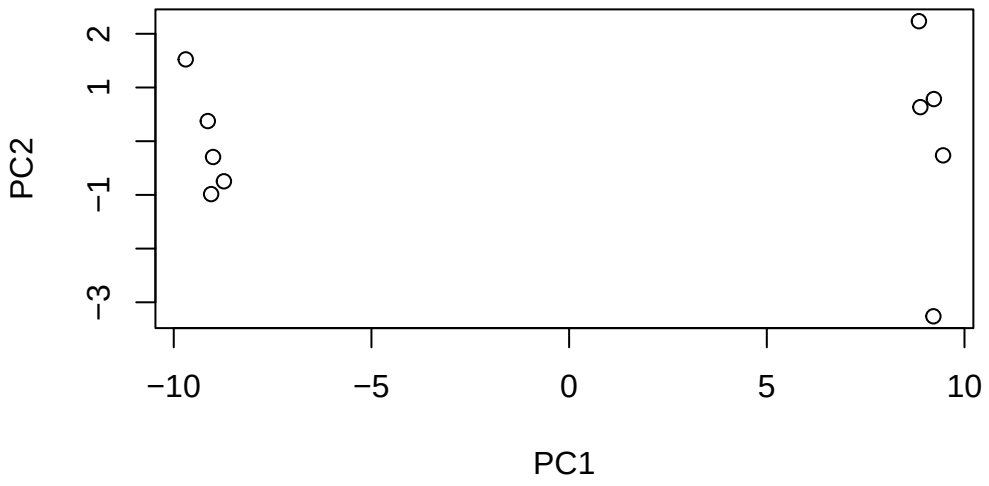
```
dim(rna.data)
```

```
[1] 100  10
```

```
#There are 100 genes and 10 samples
```

##Run PCA

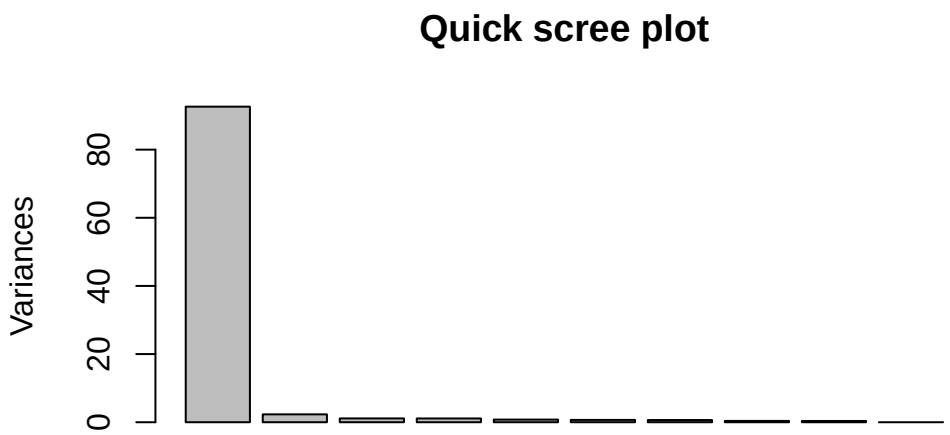
```
## Again we have to take the transpose of our data
pca <- prcomp(t(rna.data), scale=TRUE)

## Simple un polished plot of pc1 and pc2
plot(pca$x[,1], pca$x[,2], xlab="PC1", ylab="PC2")
```




```
summary(pca)
```

```
Importance of components:
                          PC1    PC2     PC3     PC4     PC5     PC6     PC7
Standard deviation     9.6237 1.5198 1.05787 1.05203 0.88062 0.82545 0.80111
Proportion of Variance 0.9262 0.0231 0.01119 0.01107 0.00775 0.00681 0.00642
Cumulative Proportion  0.9262 0.9493 0.96045 0.97152 0.97928 0.98609 0.99251
                           PC8     PC9      PC10
Standard deviation     0.62065 0.60342 3.457e-15
Proportion of Variance 0.00385 0.00364 0.000e+00
Cumulative Proportion  0.99636 1.00000 1.000e+00
```

```
plot(pca, main="Quick scree plot")
```




```
pca$x
```

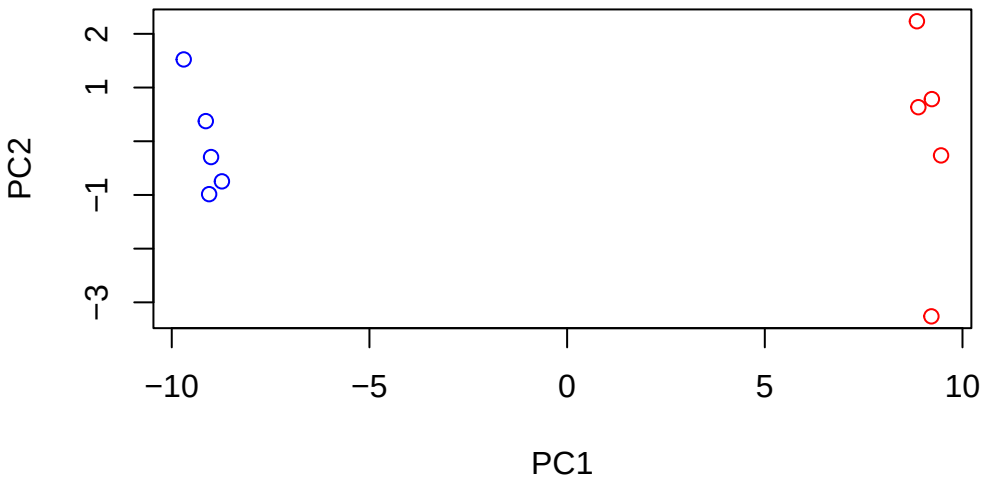
```
          PC1        PC2        PC3        PC4        PC5        PC6
wt1 -9.697374  1.5233313 -0.2753567  0.7322391 -0.6749398  1.1823860
wt2 -9.138950  0.3748504  1.0867958 -1.9461655  0.7571209 -0.4369228
wt3 -9.054263 -0.9855163  0.4152966  1.4166028  0.5835918  0.6937236
wt4 -8.731483 -0.7468371  0.5875748  0.2268129 -1.5404775 -1.2723618
wt5 -9.006312 -0.2945307 -1.8498101 -0.4303812  0.8666124 -0.2496025
```

```
ko1  8.846999  2.2345475 -0.1462750 -1.1544333 -0.6947862  0.7128021
ko2  9.213885 -3.2607503  0.2287292 -0.7658122 -0.4922849  0.9170241
ko3  9.458412 -0.2636283 -1.5778183  0.2433549  0.3654124 -0.5837724
ko4  8.883412  0.6339701  1.5205064  0.7760158  1.2158376 -0.1446094
ko5  9.225673  0.7845635  0.0103574  0.9017667 -0.3860869 -0.8186668
            PC7         PC8         PC9         PC10
wt1 -0.24446614  1.03519396  0.07010231 3.073930e-15
wt2 -0.03275370  0.26622249  0.72780448 1.963707e-15
wt3 -0.03578383 -1.05851494  0.52979799 2.893519e-15
wt4 -0.52795595 -0.20995085 -0.50325679 2.872702e-15
wt5  0.83227047 -0.05891489 -0.81258430 1.693090e-15
ko1 -0.07864392 -0.94652648 -0.24613776 4.052314e-15
ko2  0.30945771  0.33231138 -0.08786782 3.268219e-15
ko3 -1.43723425  0.14495188  0.56617746 2.636780e-15
ko4 -0.35073859  0.30381920 -0.87353886 3.615164e-15
ko5  1.56584821  0.19140827  0.62950330 3.379241e-15
```

```r
#We have 5wt and 5 ko samples, so add color
mycols <- c(rep("blue", 5), rep("red", 5))
mycols
```

```
 [1] "blue" "blue" "blue" "blue" "blue" "red"  "red"  "red"  "red"  "red"
```

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



I could examine which genes contribute most to this first PC

head(sort(abs(pca$rotation[,1]), decreasing = T))