Class07

Chen Hsiao(PID=A59026768)

Machine Learning

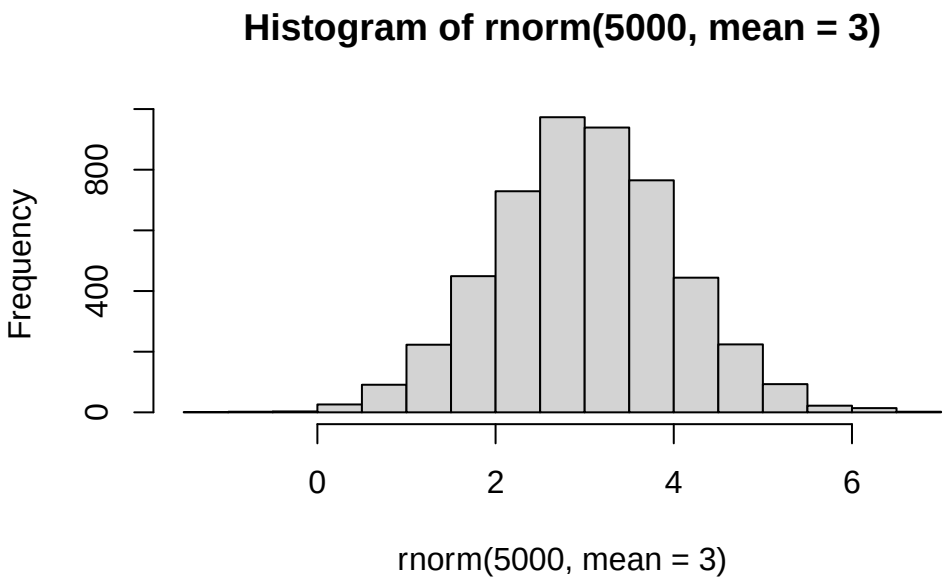
Kmeans clustering

The main function for k-means in "base" R is called 'kmeans()'. Let's first make up some data to see how k-means works and to get at the results.

```
rnorm(10)
```

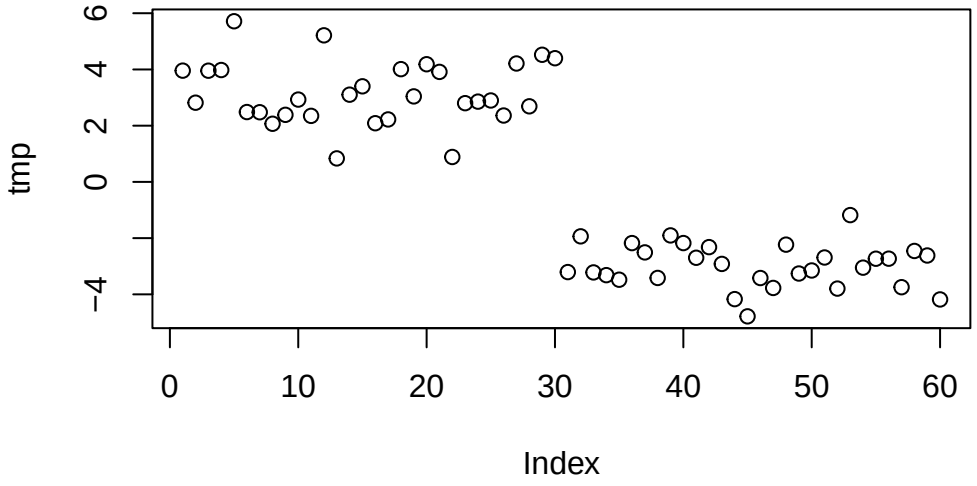
```
 [1] -0.3129040 -0.1934059 -0.3773673  1.0622338 -0.7410521  1.5702531
 [7] -1.4784274 -1.1002031  0.4937297  0.3968504
```

```
hist(rnorm(5000, mean=3))
```

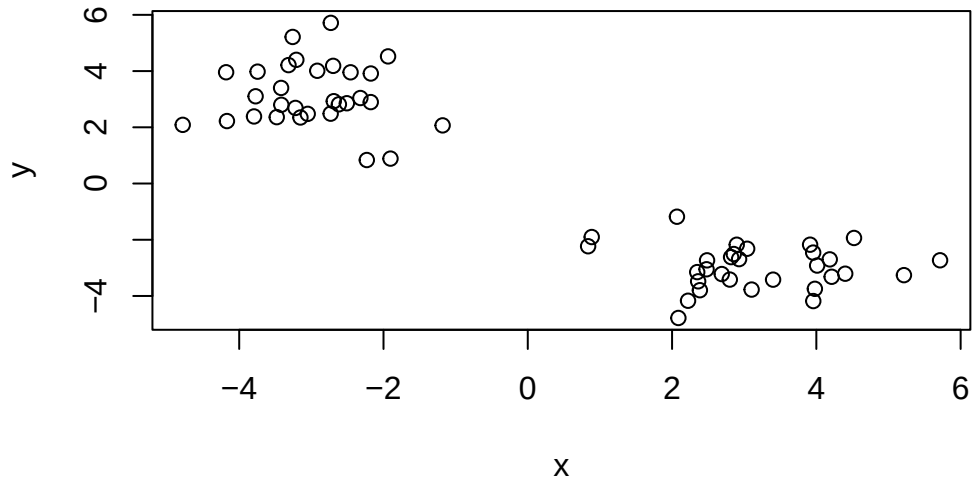


Make a wee vector with 60 total points half centered at +3 and another centered @ -3

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




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



In this set of data, we determine two groups, one is around +3 and another is -3. Let's see how kmean clusters 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  3.158578 -2.974832
2 -2.974832  3.158578

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
[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 2 2 2

Within cluster sum of squares by cluster:
[1] 54.55054 54.55054
 (between_SS / total_SS =  91.2 %)

Available components:

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

To know what is inside the list.

```
attributes(k)
```

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

$class
[1] "kmeans"
```

```
k$centers
```

```
          x         y
1  3.158578 -2.974832
2 -2.974832  3.158578
```

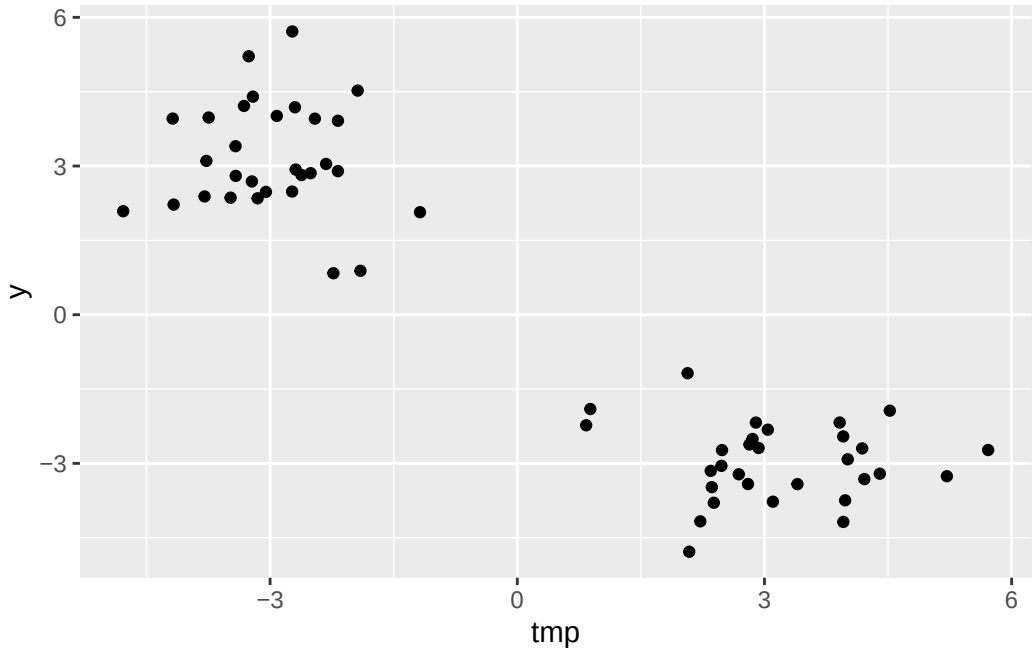
What is my cluster result?

```
k$cluster
```

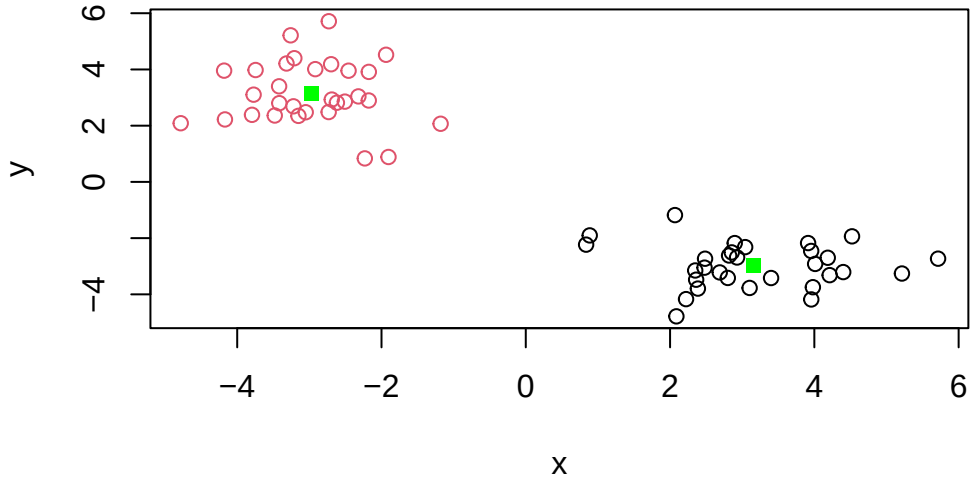
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Q. Plot your data 'x' showing ur clustering result and center point for each cluster.

```
library(ggplot2)
ggplot(data.frame(x), aes(x = tmp, y = y))+
  geom_point()
```

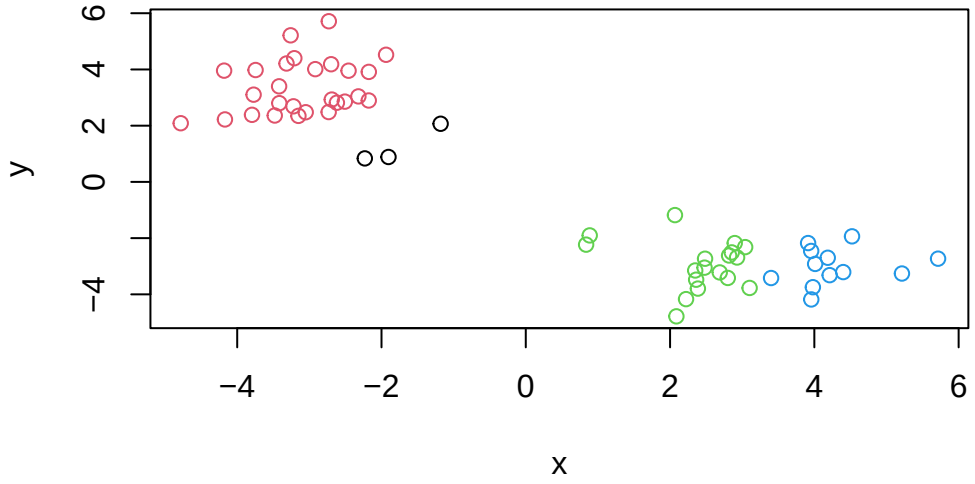



```
plot(x, col=k$cluster)
points(k$centers, pch = 15, col = "green")
```



Q. Run kmeans and clster into 3 group and lot the result.

```
k4 <- kmeans(x, centers = 4)
plot(x, col = k4$cluster)
```



From k=2 to k=3, you'll find out an elbow at 3. This is the

```
k$tot.withinss
```

```
[1] 109.1011
```

```
#k3$tot.withinss
```

The big limmitation of kmeans is that it imposes a structure on ur data (i.e. a clustering) that you ask for in the first place.

Hierarchical Clustering

The main function in "base" R for this is called 'hclust()'. It wants a distance matrix as input no the data itself.

We can calculate a distance matrix in lots of different ways but here we will use the 'dist( )' function, which calculate euclidean distance

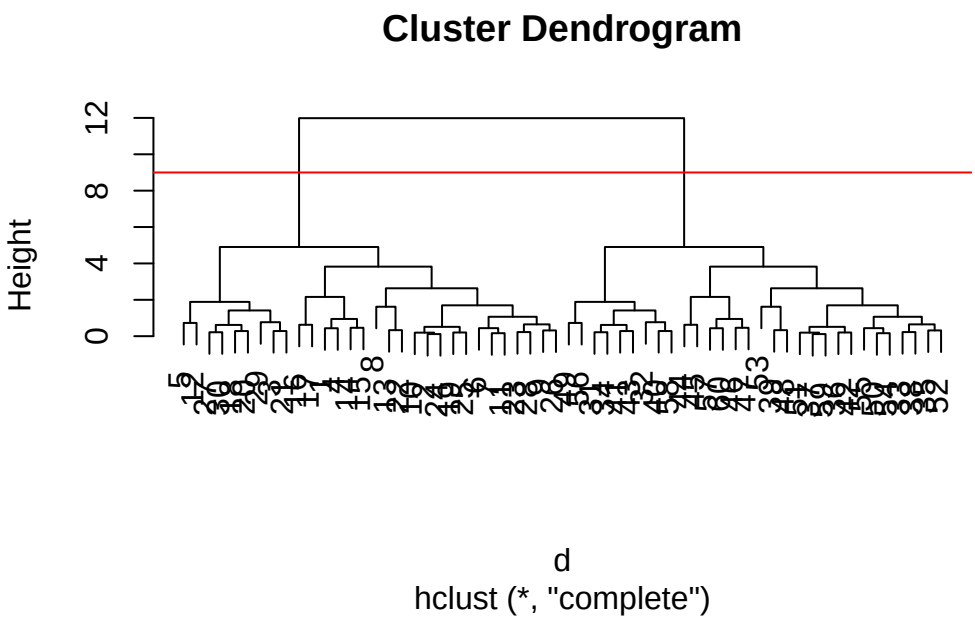
```
d <- dist(x, diag = T)
hc <- hclust(d)
hc
```

```
Call:
hclust(d = d)

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

There is a specific plot method

```
plot(hc)
abline(h = 9, col = "red")
```

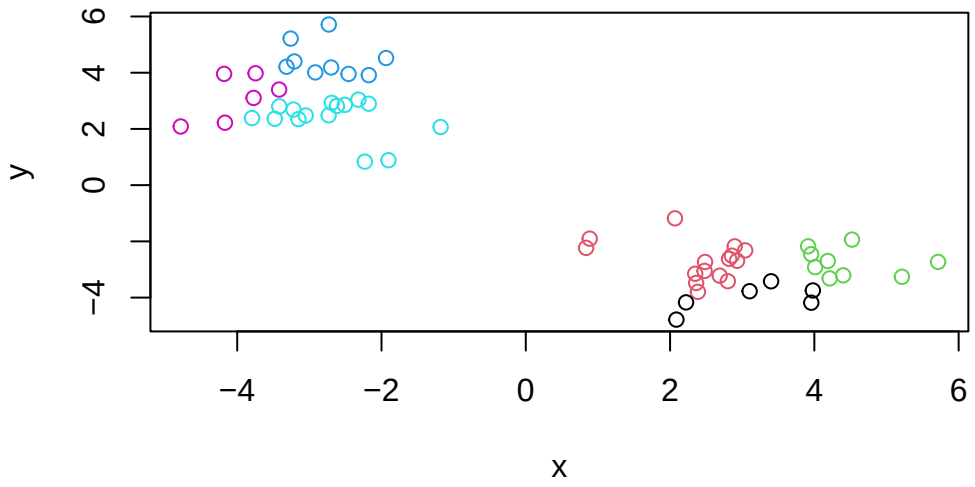


To get the cluster membership vector we need to "cut" the tree at a given height that we pick. The function to do this is called 'cutree()'.

```
cutree(hc,h = 3)
```

```
 [1] 1 2 3 1 3 2 2 2 2 2 2 3 2 1 1 1 1 3 2 3 3 2 2 2 2 2 3 2 3 3 4 4 5 4 5 5 5 5
[39] 5 4 4 5 4 6 6 6 6 5 4 5 5 5 5 5 5 4 6 4 5 6
```

```
plot(x, col = cutree(hc,h = 3))
```

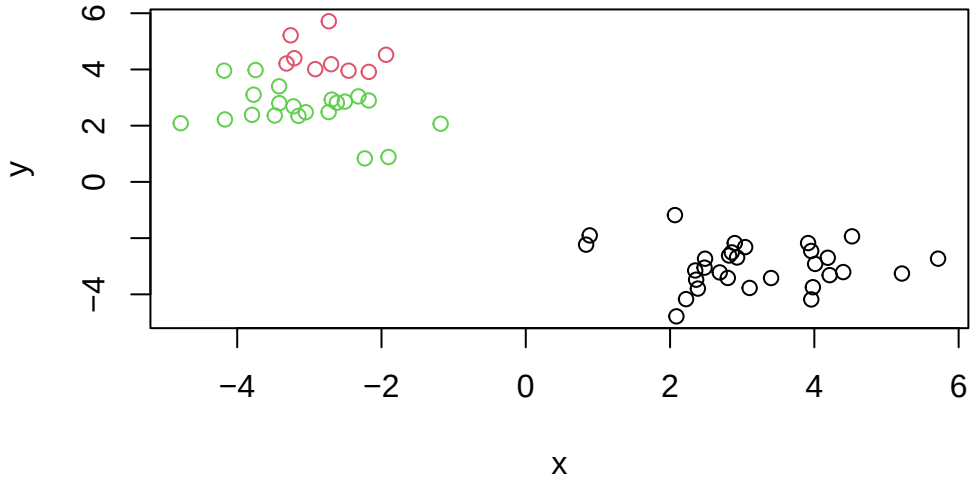



```
grps <- cutree(hc, k=3)
grps
```

```
 [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 3 2 3 3 3 3
[39] 3 2 2 3 2 3 3 3 3 3 2 3 3 3 3 3 3 2 3 2 3 3
```

Q. Plot our data ('x') colored by our hclust result.

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



UK-food

PCA: create a new vector which describe the variance the most

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

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

```
nrow(x)
```

```
[1] 17
```

```
ncol(x)
```

```
[1] 5
```

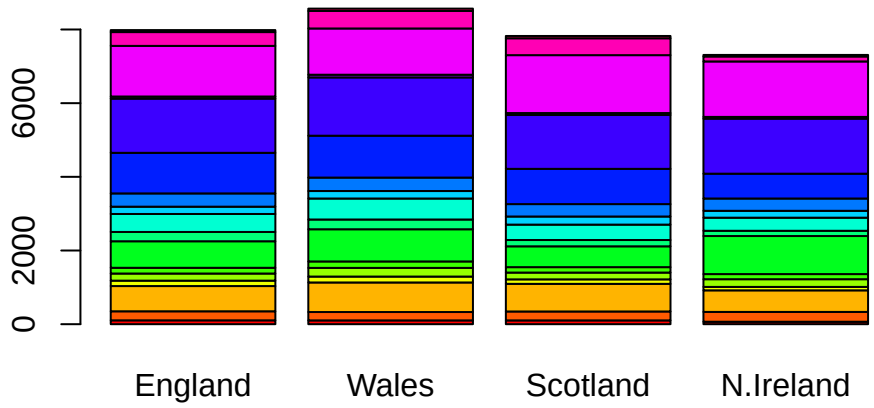
```
x <- read.csv(url, row.names=1)
```

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 one because we only need one command line to name the row. Using x<-x[,-1] sometimes may accidently delete the coloumn you want.

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

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

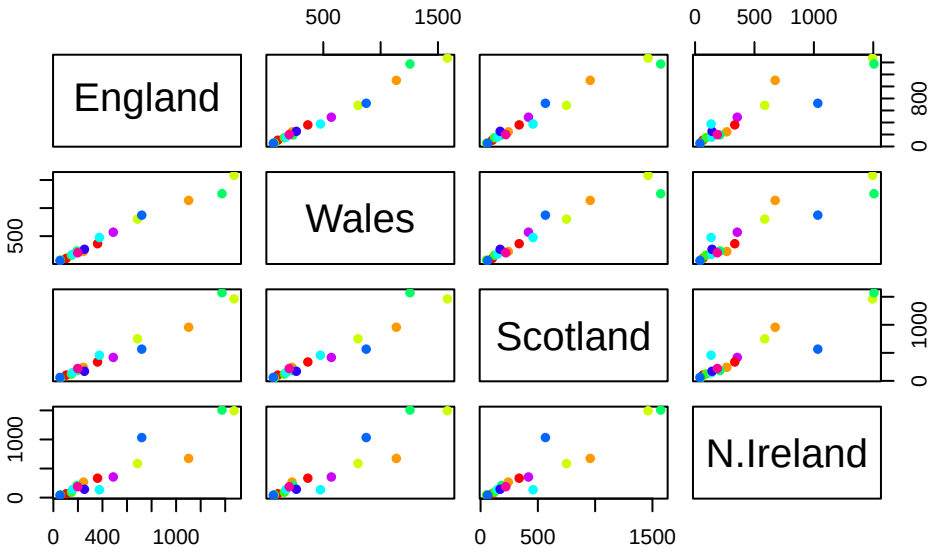



```
#legend("topright", legend = rownames(x), fill = rainbow(10), title = "Legend")
```

One useful plot in this case (because we only have 4 countries to look across) is pairing.

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?

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



For a given plot, if points lie on the diagonal, it means that the two regions the plot is comparing show similar food preference.

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

Fresh_veg is the main differences between N. Ireland and other countries.

Enter PCA

The main function to do PCA in "base" R is call 'prcomp()'. It wants our foods as the columns and the countries as the rows. It basically want the transpose of the data we have.

```
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
```

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

```
pca$x
```

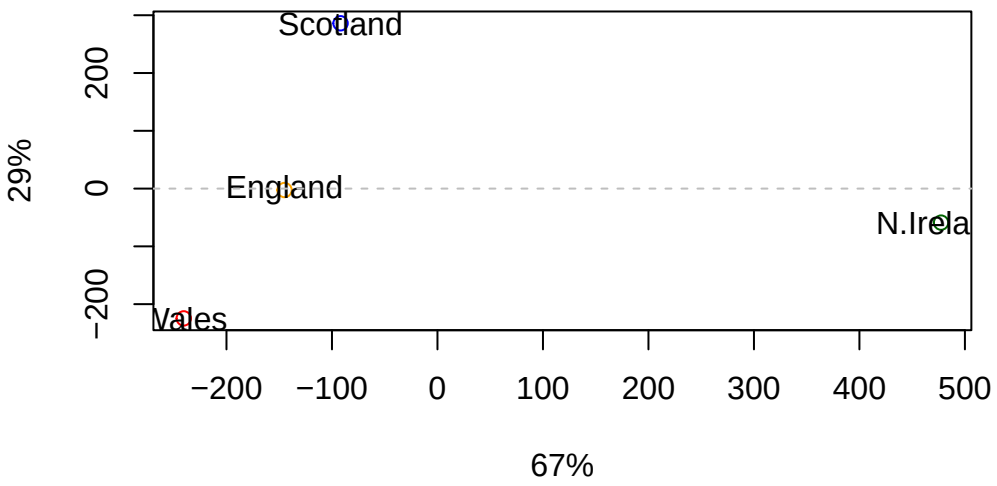
```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -4.894696e-14
Wales     -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland   -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland  477.39164  -58.901862  -4.877895  2.321303e-13
```

```
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
```

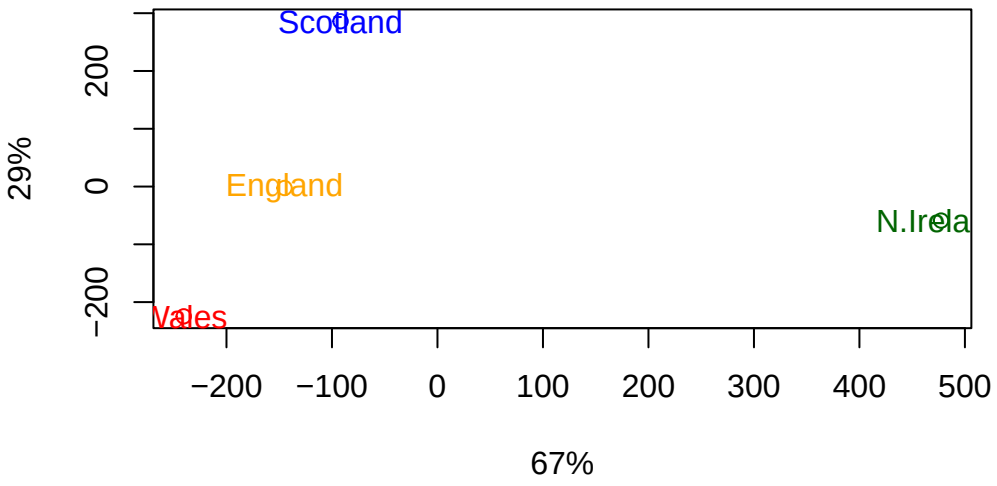
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
plot(pca$x[,1],pca$x[,2], xlab = "67%", ylab = "29%", col = c("orange", "red", "blue", "da
text(pca$x[,1], pca$x[,2], colnames(x))
abline(h=0, col= "gray", lty = 2)
```



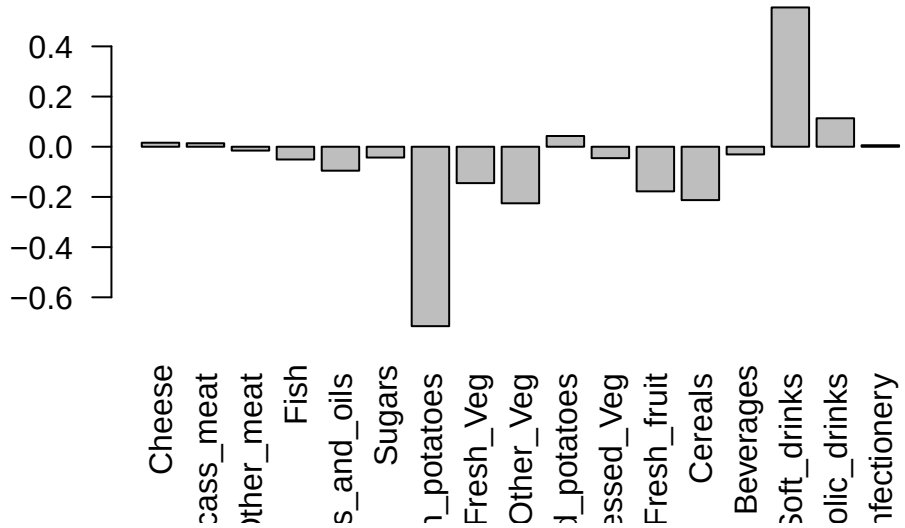
Q8. Customize your plot so that the colors of the country names match the colors in our UK and Ireland map and table at start of this document.

```
plot(pca$x[,1],pca$x[,2], xlab = "67%", ylab = "29%", col = c("orange", "red", "blue", "da
text(pca$x[,1], pca$x[,2], colnames(x), col = c("orange", "red", "blue", "darkgreen" ) )
```



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

```
barplot( pca$rotation[,2], las=2 )
```



PCA of RNA-seq data

```
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?

```
nrow(rna.data)
```

```
[1] 100
```

```
ncol(rna.data)
```

```
[1] 10
```