Class08

Natasha (PID: A15393874)

10/21/2021

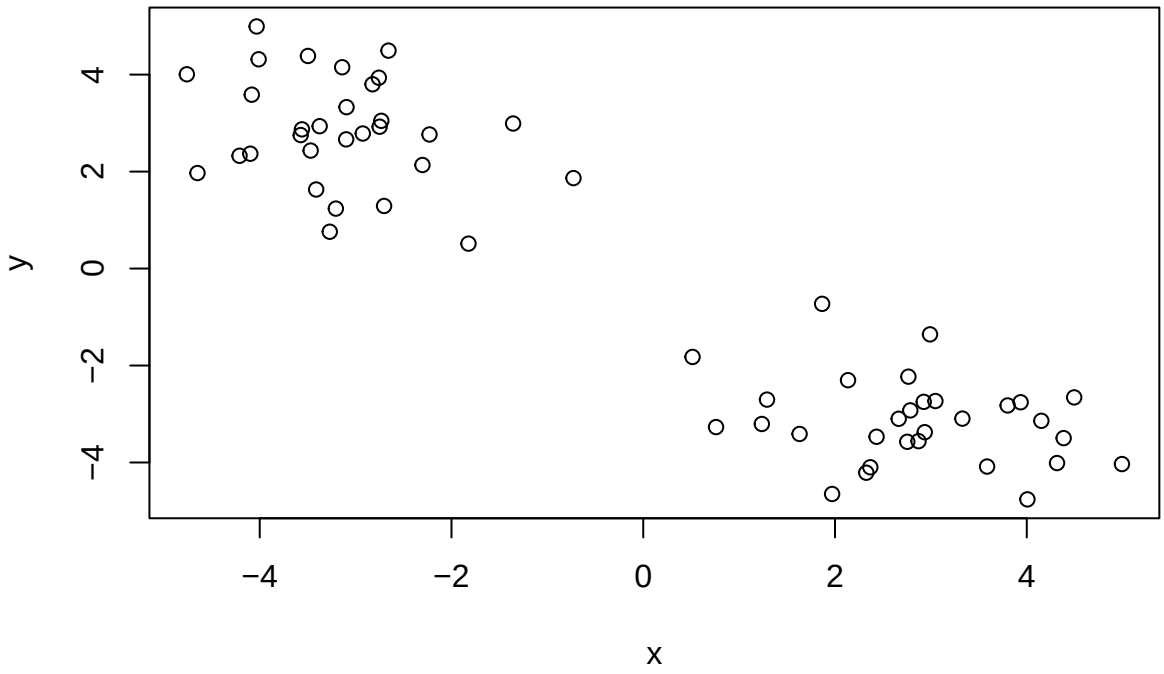
irst up is clustering methods

Kmeans clustering

The function in base R to do Kmeans clustering is called `kmeans()`.

First make up some data where we know what the answer should be:

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



> Q. Can we use kmeans() to cluster this data setting k 2 and nstart to 20?

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

```
## K-means clustering with 2 clusters of sizes 30, 30
##
## Cluster means:
##           x         y
## 1 -3.144188  2.842216
## 2  2.842216 -3.144188
##
## 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 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] 60.55669 60.55669
##  (between_SS / total_SS =  89.9 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Q. How many point are in each cluter?

```
km$size
```

```
## [1] 30 30
```

Q. What 'component' of your result object details cluster assignment/membership?

```
km$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 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
```

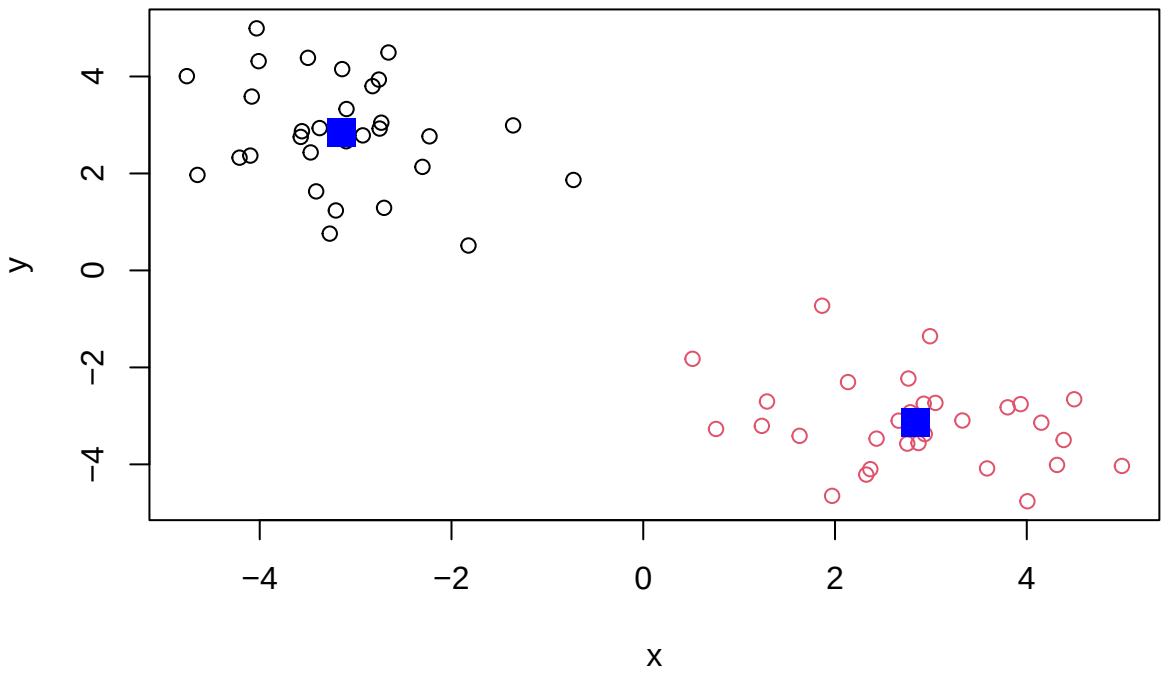
Q. What 'component' of your result object details cluster center?

```
km$centers
```

```
##           x         y
## 1 -3.144188  2.842216
## 2  2.842216 -3.144188
```

Q. Plot x colored by the kmeans cluster assignment an add cluster centers as blue points

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



Hierachial Clustering

A big limitation with k-means is that we have to tell it K (the number of clusters we want).

Analyze this same data with hclust()

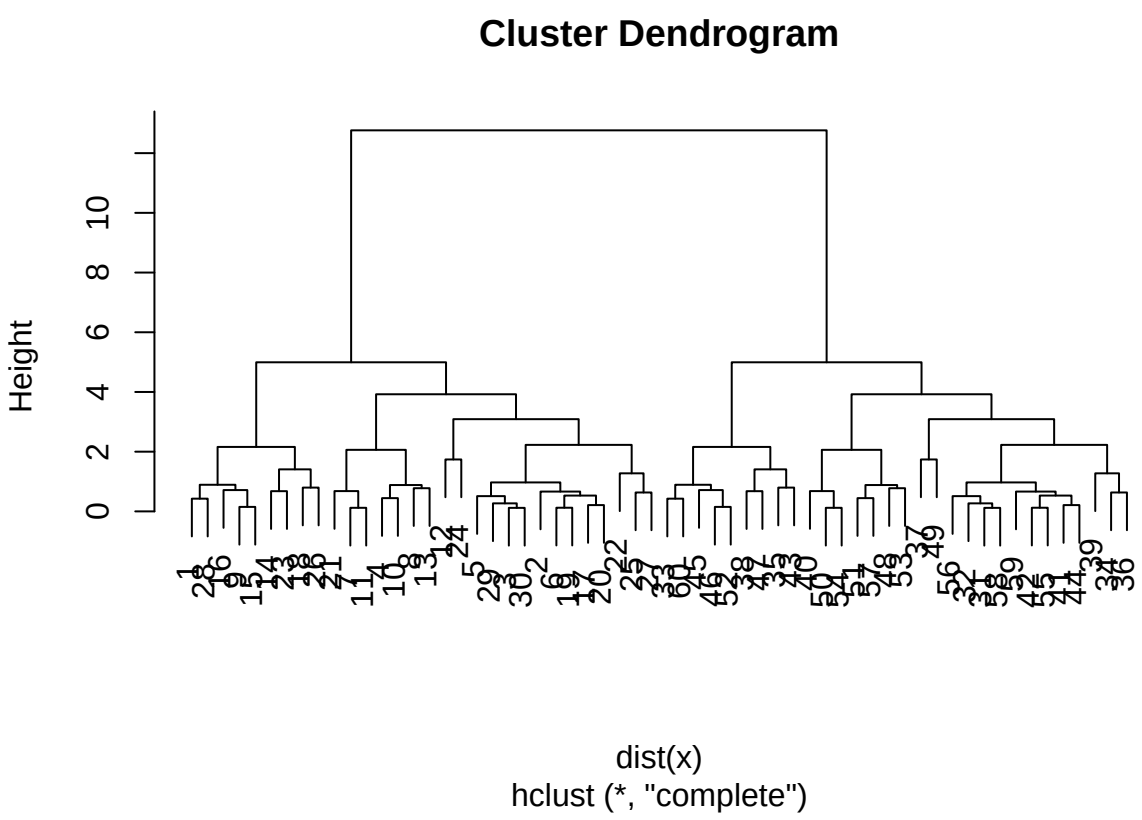
Demonstrate the use of dist(), hclust(), plot() and cutree() functions to do clustering, Generate dendrograms and return cluster assignment/membership vector...

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

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

There is a plot method for hclust result objects. Let's see it.

```
plot(hc)
```



To get our cluster membership vector we have to do a little bit more work. We hae to "cut" the tree where we think it makes sense. For this we use the`cutree()` function.

```
cutree(hc, h=6)
```

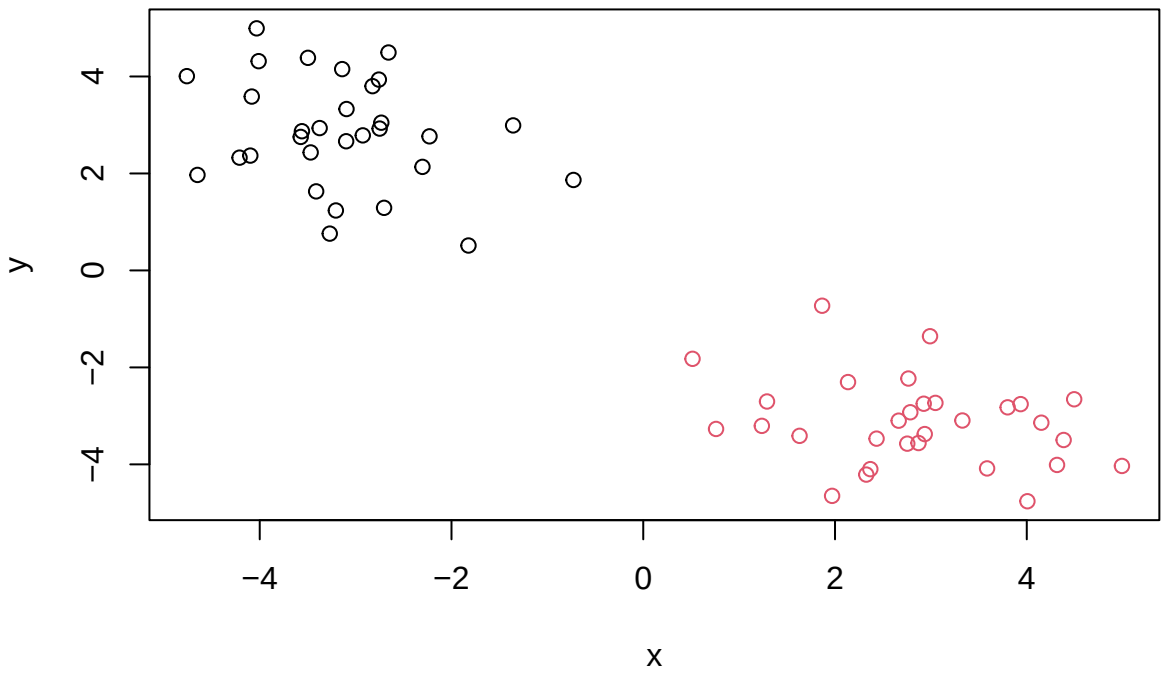
```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
## [39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

You can also call `cutree()` setting k= the numver of grps/clusters you want.

```
grps <- cutree(hc, k=2)
```

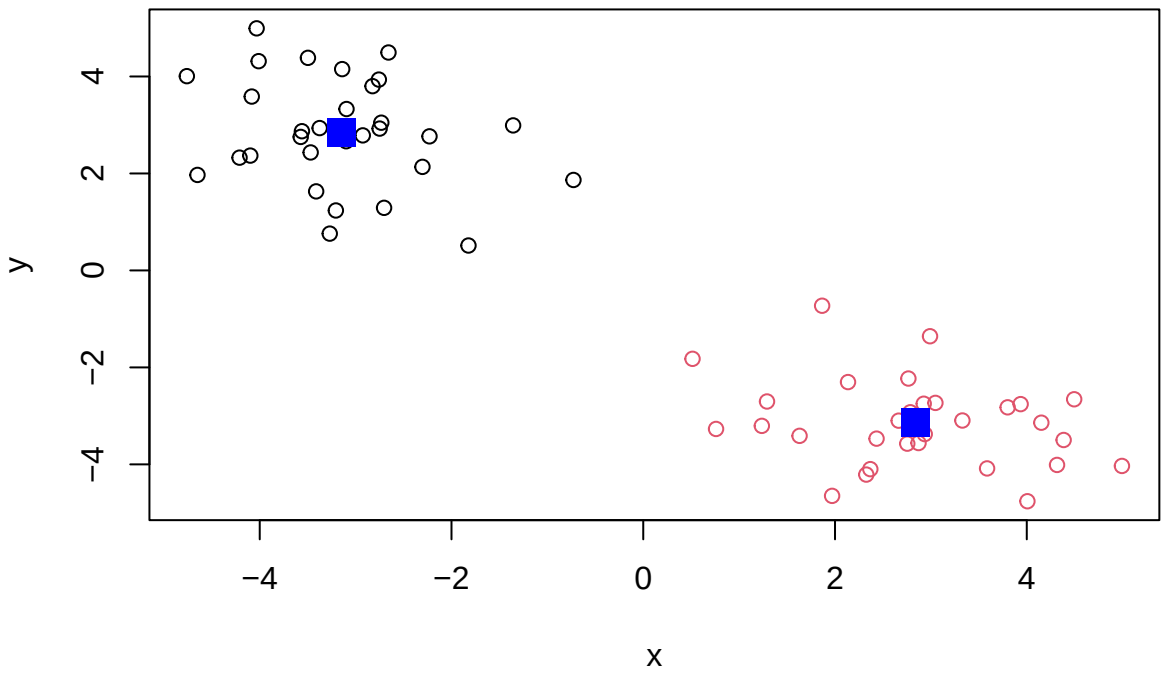
Make our results plot

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



Q. Plot x colored by the kmeans cluster assignment an add cluster centers as blue points

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




```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url)
```

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); ncol(x); nrow(x)
```

```
## [1] 17  5
```

```
## [1] 5
```

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

Preview the first 6 rows

```
View(x)
```

```
# 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
```

```
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
```

```
dim(x)
```

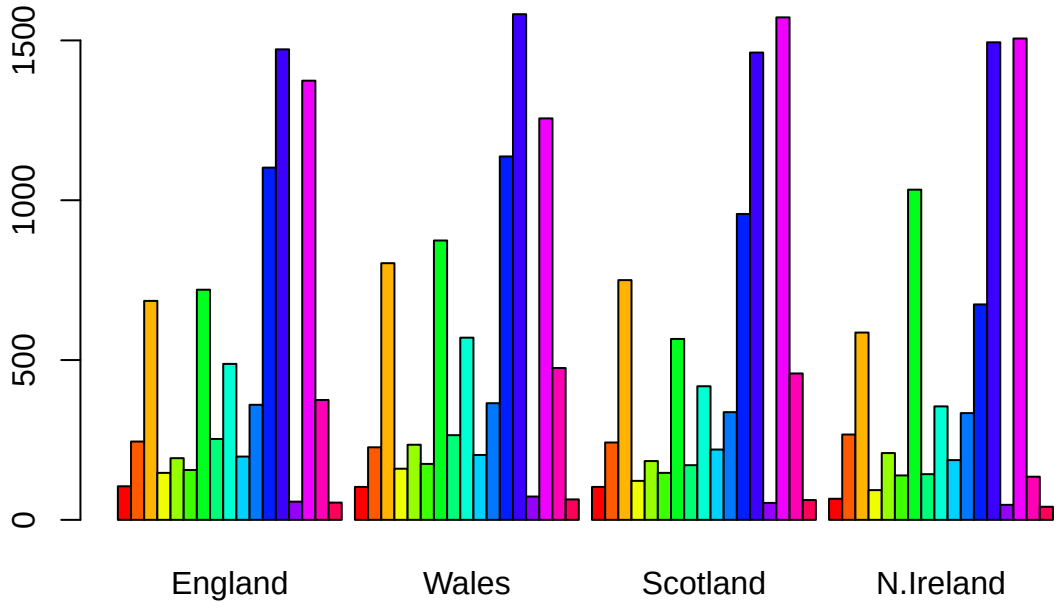
```
## [1] 17  4
```

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

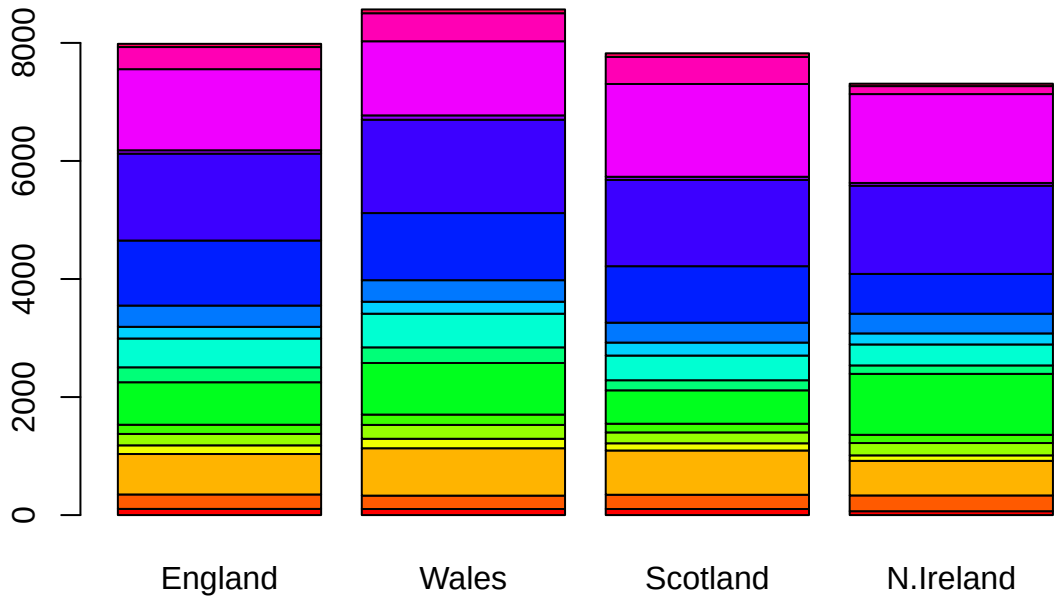
```
##                England Wales Scotland N.Ireland
## Cheese             105   103      103        66
## Carcass_meat       245   227      242       267
## Other_meat         685   803      750       586
## Fish               147   160      122        93
## Fats_and_oils      193   235      184       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 one becase if you keep running the x <- x[,-1] a couple of times it removes rows.

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



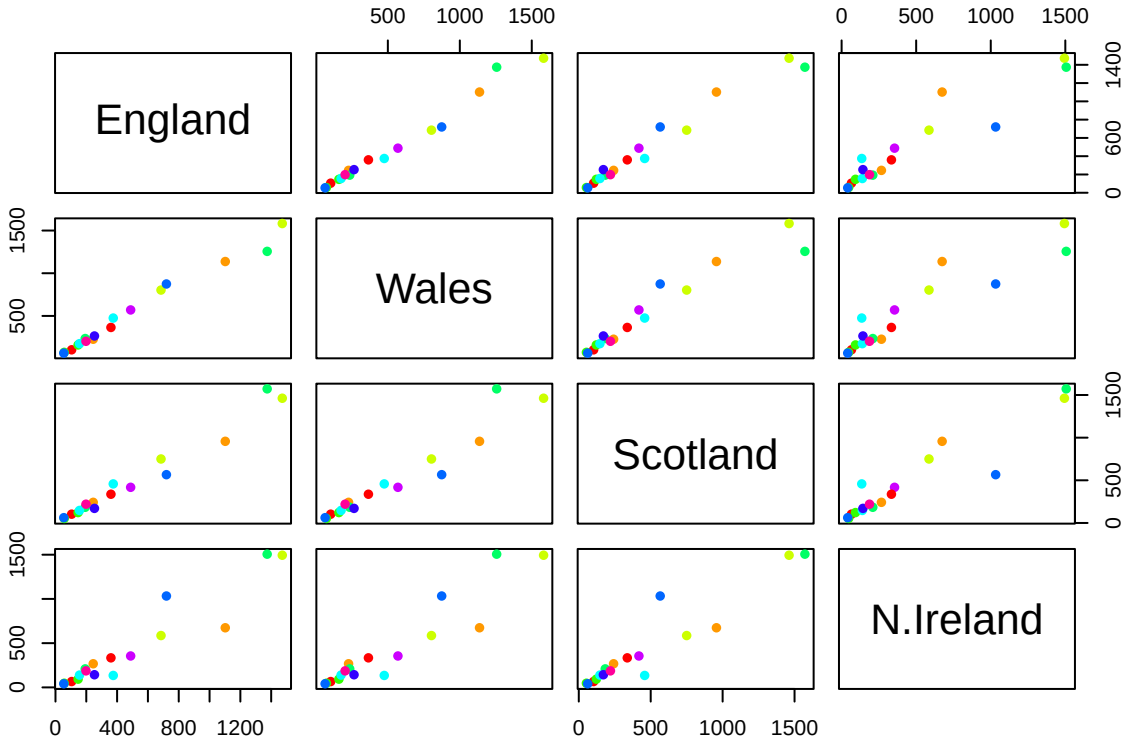

```
barplot(as.matrix(x), beside=FALSE, col=rainbow(nrow(x)))
```



Q3: Changing what optional argument in the above barplot() function results in the following plot? Changing besides=false changes the appearance

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 two countries vs the other country so comparing the two countries. If the values are exactly the same amount of food the points lie on the diagonal. So lying on the diagonal means similar values.

```
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? You cannot tell.

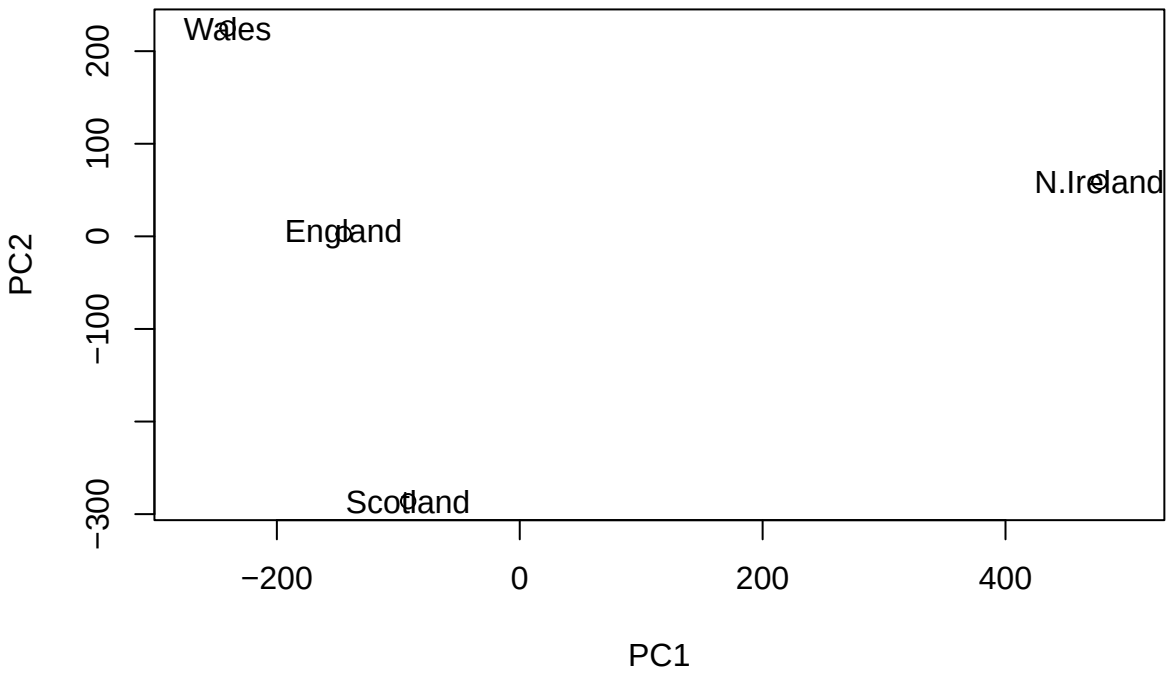
PCA to the rescue!

```
# Use the prcomp() PCA function
pca <- prcomp( t(x) )
summary(pca)
```

```
## Importance of components:
##                             PC1      PC2      PC3        PC4
## Standard deviation     324.1502 212.7478 73.87622 4.189e-14
## Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
## Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

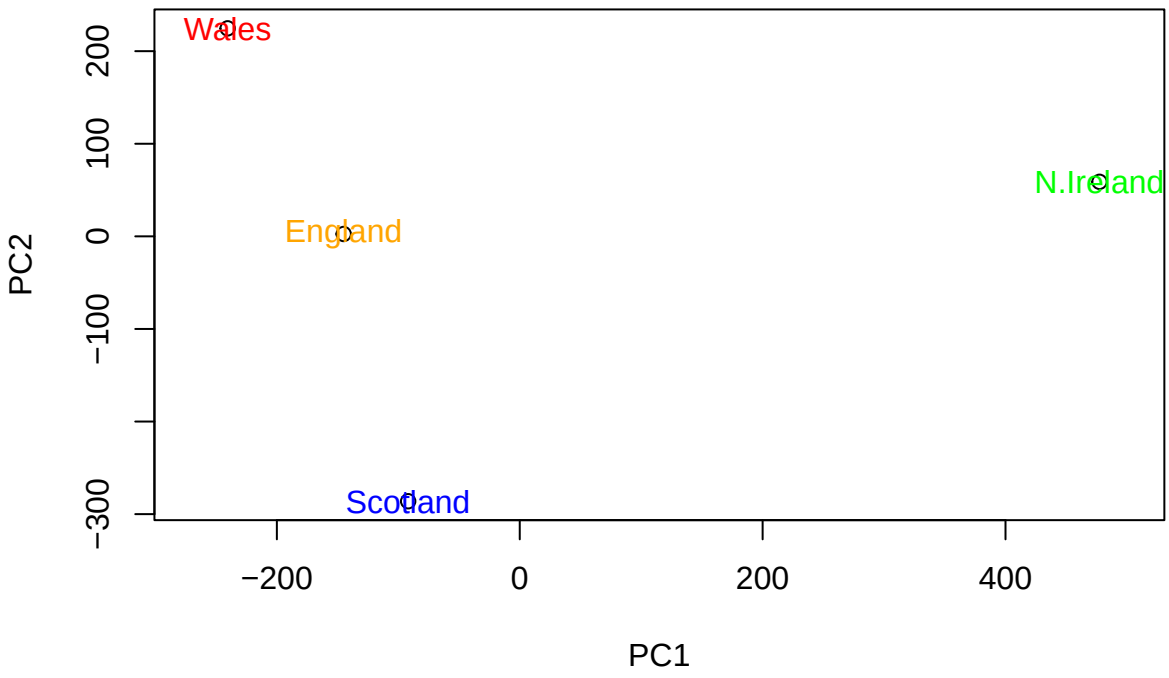
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
# 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.

```
# Plot PC1 vs PC2
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)
```



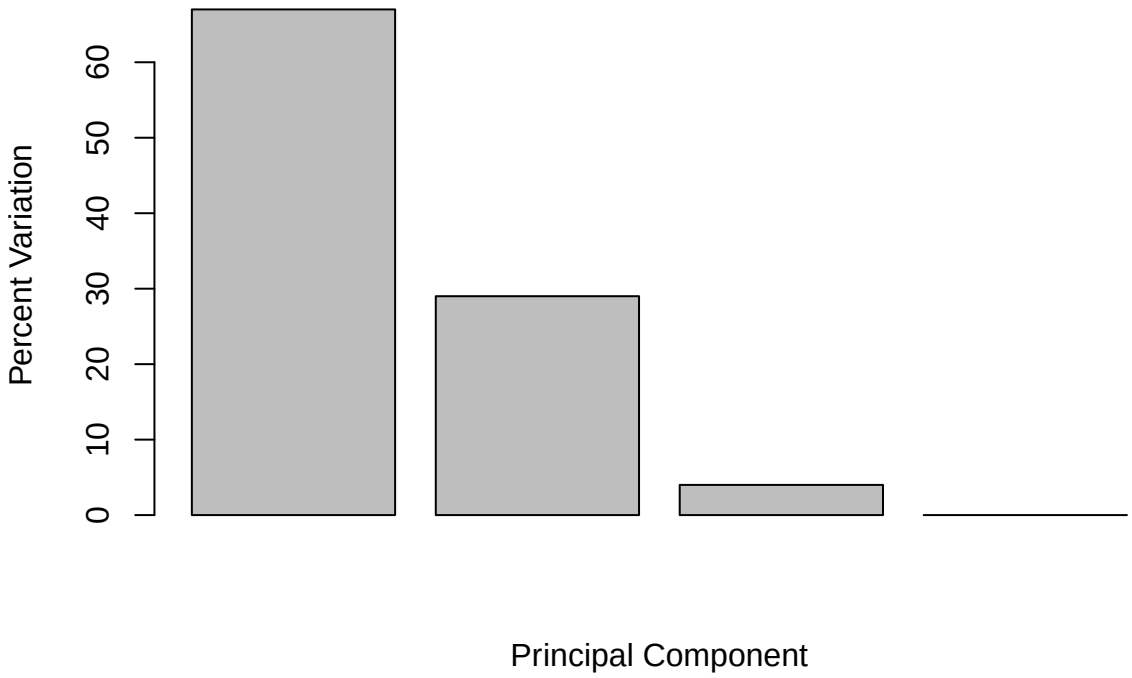

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

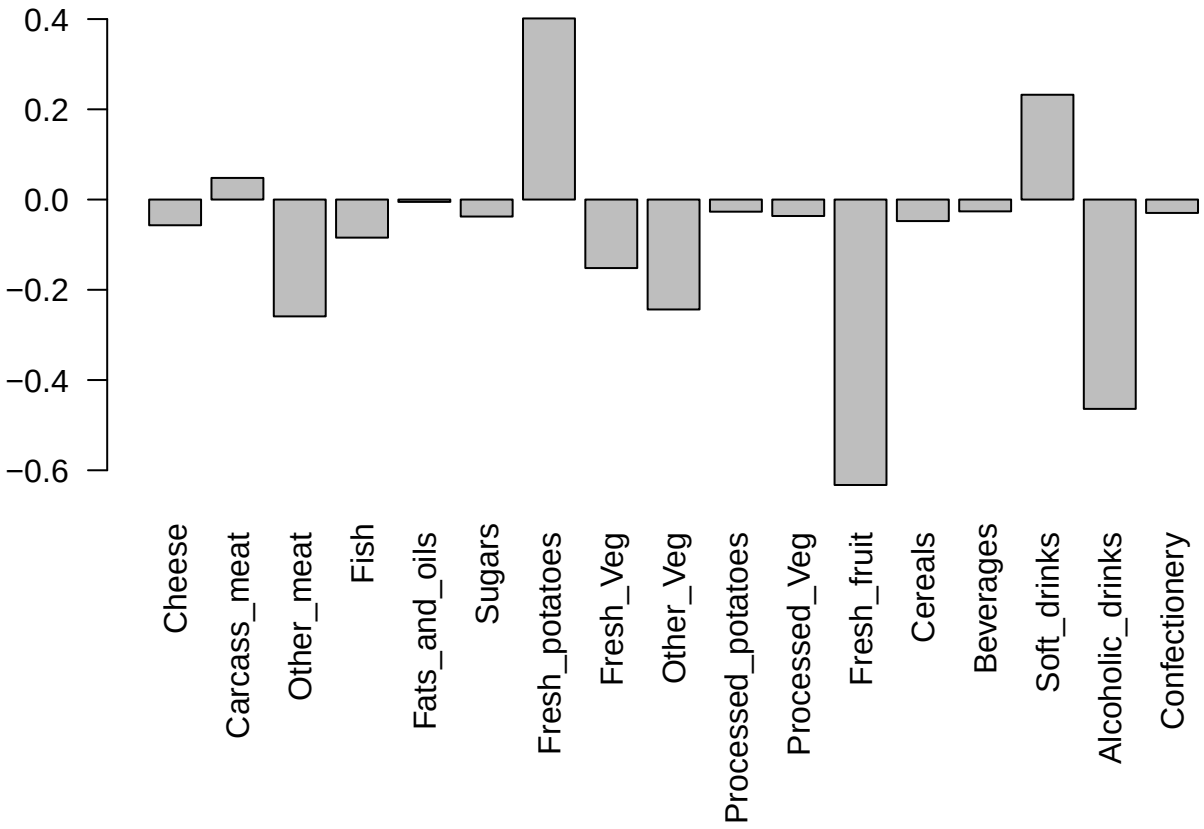
```
## or the second row here...
z <- summary(pca)
z$importance
```

```
##                             PC1       PC2      PC3          PC4
## Standard deviation     324.15019 212.74780 73.87622 4.188568e-14
## Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
## Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

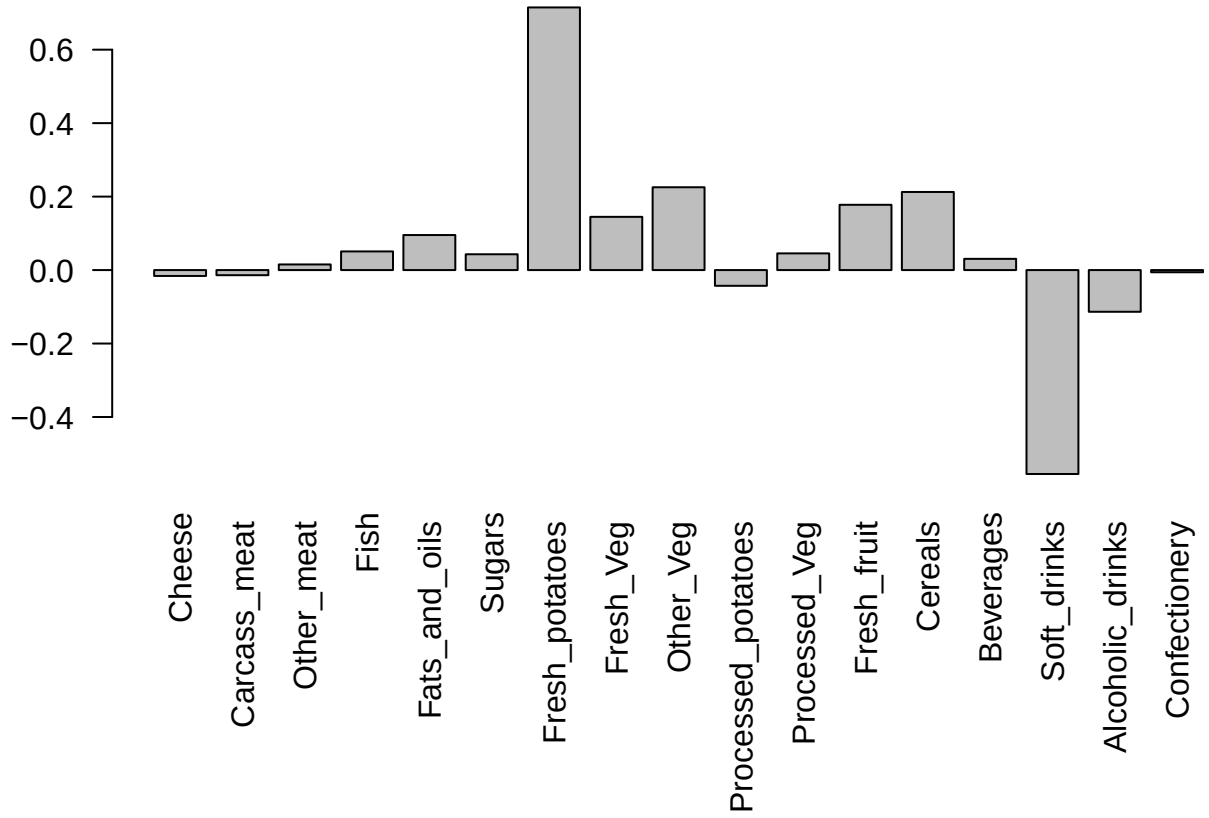
```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```




```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,1], las=2 )
```




```
## Lets focus on PC1 as it accounts for > 90% of variance
par(mar=c(10, 3, 0.35, 0))
barplot( pca$rotation[,2], las=2 )
```



> Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 maninly tell us about? Fresh potatoes and soft drinks are featured the most prominently. This tells us that there is great variation comparing Ireland to other countries in the consumption of these foods.

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

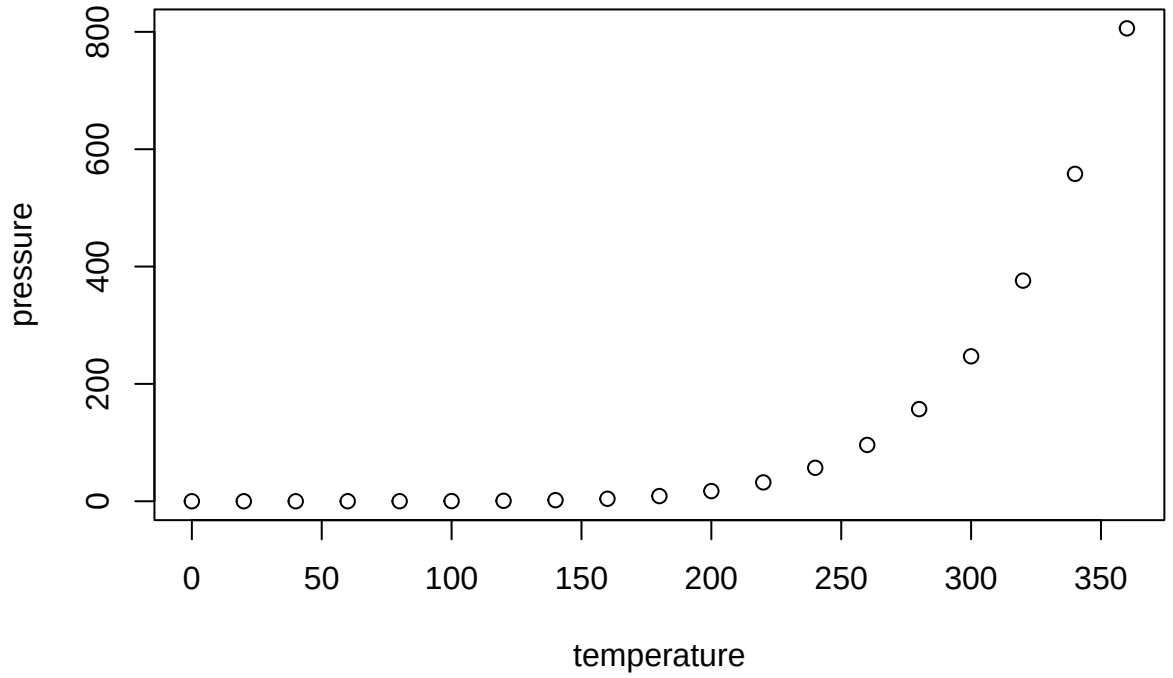
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.