class07

Christina Mac

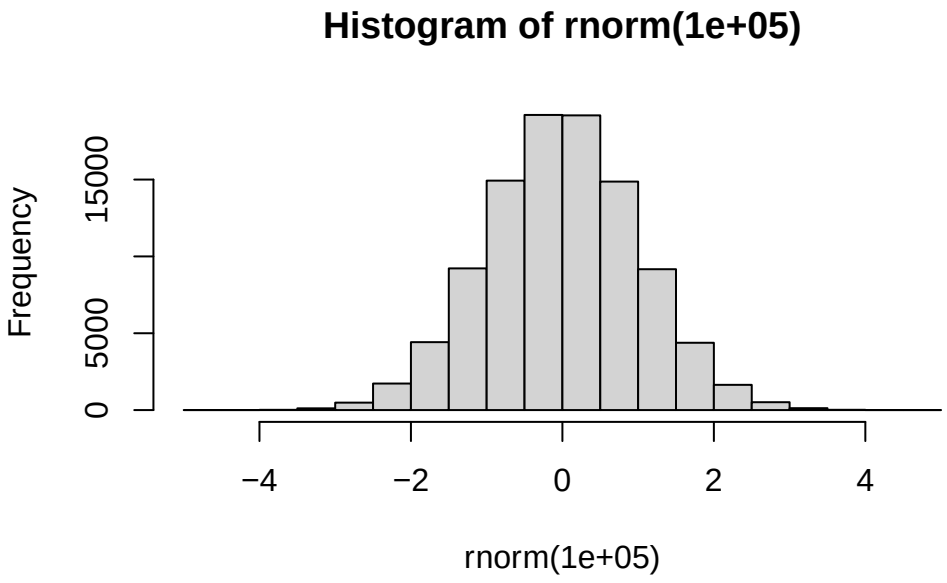
In this class, we will explore clustering and dimensionality reduction methods.

##K-means Make up some input data where we know what the answer should be.

```
#will give 10 random values around 0
rnorm(10)
```

```
 [1]  0.1897638 -2.0826403 -0.5414565  0.1074026  0.6090812  0.9567687
 [7] -1.2423187  0.3573621  0.3276948  0.9769798
```

```
#give the histogram for the random numbers
hist(rnorm(100000))
```



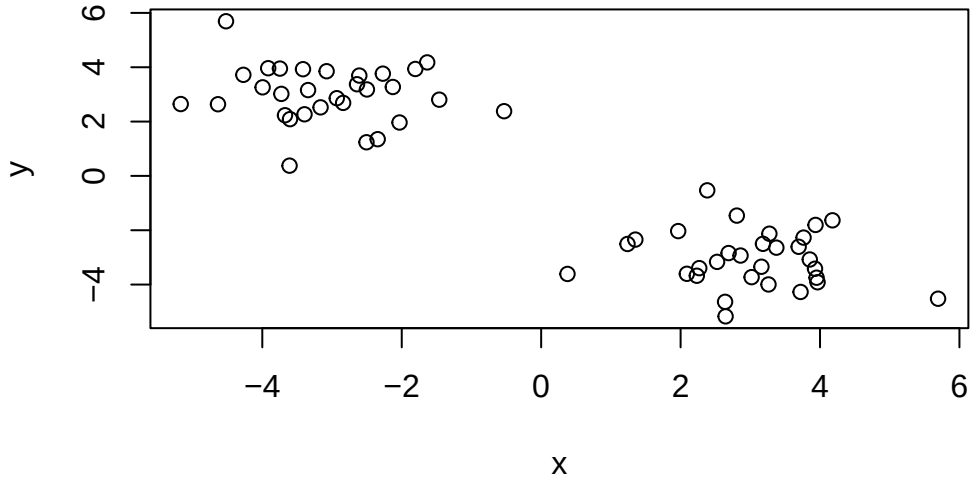
```r
#the default mean is 0, you can change that
#making a vector
temp <-c(rnorm(30,-3), rnorm(30,+3))
#reverse --> will switch the two vectors in temp
x<-cbind(x=temp, y=rev(temp))
x
```

```
               x          y
 [1,] -2.9316783  2.8598715
 [2,] -3.7472245  3.9505221
 [3,] -1.6345430  4.1784898
 [4,] -4.2722115  3.7209439
 [5,] -3.3436294  3.1594931
 [6,] -2.6084867  3.6934565
 [7,] -1.4615995  2.8065138
 [8,] -3.6747985  2.2328705
 [9,] -2.1275243  3.2737126
[10,] -3.6028485  2.0897154
[11,] -0.5318571  2.3821544
[12,] -3.7271953  3.0186262
[13,] -3.1643301  2.5234309
[14,] -5.1702834  2.6452740
[15,] -2.5054747  1.2385806
[16,] -2.8405734  2.6886219
[17,] -3.3946443  2.2679400
[18,] -2.4994445  3.1817357
[19,] -2.2710892  3.7638057
[20,] -2.3450044  1.3510026
[21,] -2.0317188  1.9658807
[22,] -3.9156468  3.9648640
[23,] -3.9954210  3.2608736
[24,] -4.6358055  2.6380605
[25,] -3.4165330  3.9280370
[26,] -3.6093445  0.3773269
[27,] -4.5208801  5.6934195
[28,] -1.8051465  3.9357690
[29,] -2.6412644  3.3739524
[30,] -3.0775085  3.8529850
[31,]  3.8529850 -3.0775085
[32,]  3.3739524 -2.6412644
[33,]  3.9357690 -1.8051465
[34,]  5.6934195 -4.5208801
```

```
[35,]  0.3773269 -3.6093445
[36,]  3.9280370 -3.4165330
[37,]  2.6380605 -4.6358055
[38,]  3.2608736 -3.9954210
[39,]  3.9648640 -3.9156468
[40,]  1.9658807 -2.0317188
[41,]  1.3510026 -2.3450044
[42,]  3.7638057 -2.2710892
[43,]  3.1817357 -2.4994445
[44,]  2.2679400 -3.3946443
[45,]  2.6886219 -2.8405734
[46,]  1.2385806 -2.5054747
[47,]  2.6452740 -5.1702834
[48,]  2.5234309 -3.1643301
[49,]  3.0186262 -3.7271953
[50,]  2.3821544 -0.5318571
[51,]  2.0897154 -3.6028485
[52,]  3.2737126 -2.1275243
[53,]  2.2328705 -3.6747985
[54,]  2.8065138 -1.4615995
[55,]  3.6934565 -2.6084867
[56,]  3.1594931 -3.3436294
[57,]  3.7209439 -4.2722115
[58,]  4.1784898 -1.6345430
[59,]  3.9505221 -3.7472245
[60,]  2.8598715 -2.9316783
```

Quick plot of x to see the two groups at -3,+3 and +3,-3.

```
plot(x)
```



Use the `kmeans()` function setting k to 2 and nstart=20

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

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

Cluster means:
          x         y
1 -3.050124  3.000598
2  3.000598 -3.050124

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

Within cluster sum of squares by cluster:
[1] 62.65619 62.65619
 (between_SS / total_SS =  89.8 %)

Available components:
```

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

Inspect the results >How many points are in each cluster?

```
km$size
```

```
[1] 30 30
```

What component of you result object details: - cluster assignment/membership? - cluster center?

```
km$cluster
```

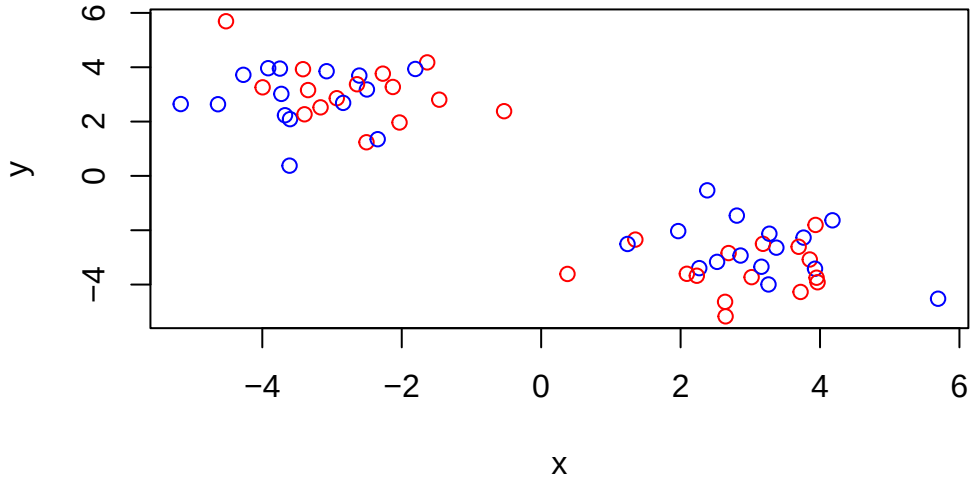
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
km$centers
```

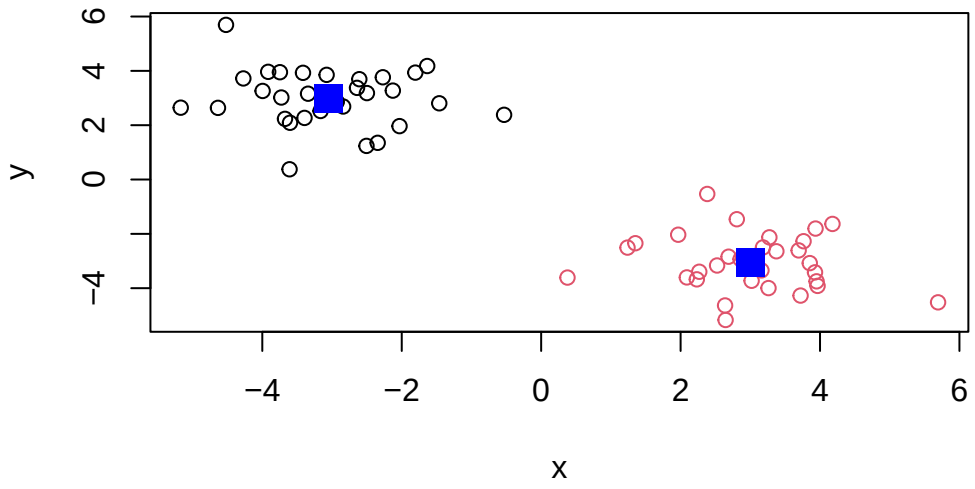
```
          x         y
1 -3.050124  3.000598
2  3.000598 -3.050124
```

Plot x colored by the kmeans cluster assignment and add cluster centers as blue points

```
#this plot will color alternating points, does not really tell you much
plot(x, col=c("red", "blue"))
```

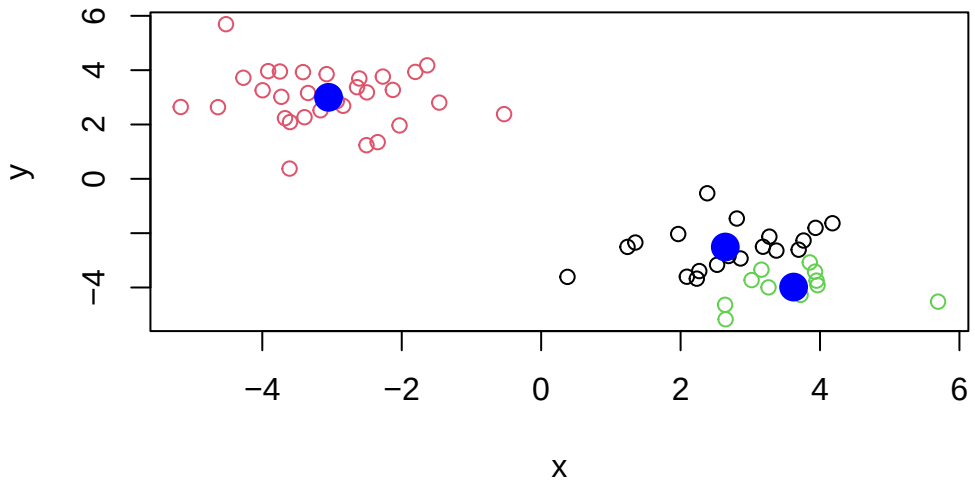



```
plot(x, col=km$cluster)
points(km$centers, col = "blue", pch=15, cex = 2)
```



Play with kmeans and ask fo different number of clusters

```
km <-kmeans(x, centers=3, nstart=20)
plot(x, col=km$cluster)
points(km$centers, col="blue", pch=16, cex = 2 )
```



Hierarchical clustering

This is another very useful and widely employed clustering method which has the advantage over kmeans in that it can help reveal the something of th true groups in your data.

The `hclust()` function wants a distance matrix as input. We an get this from the `dist()` function.

```
d <- dist(x)
#wants a distance matrix, any type
hc <- hclust(d)
hc
```

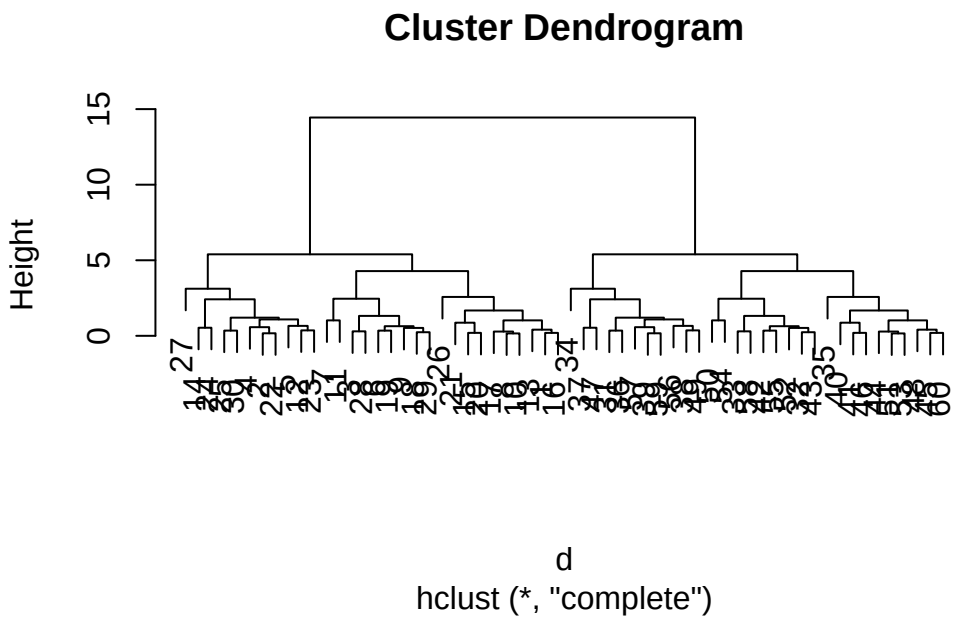
```
Call:
```

```
hclust(d = d)

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

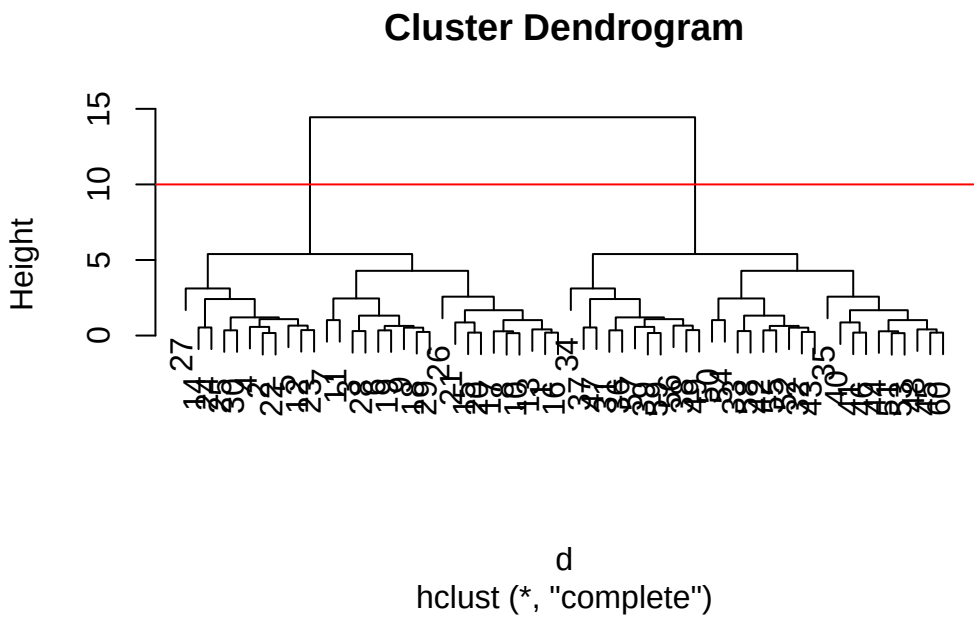
There is a plot method for hclust results:

```
plot(hc)
```



Branch length (height) is proportional to how "close" the two points are. Closer points are grouped together.

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

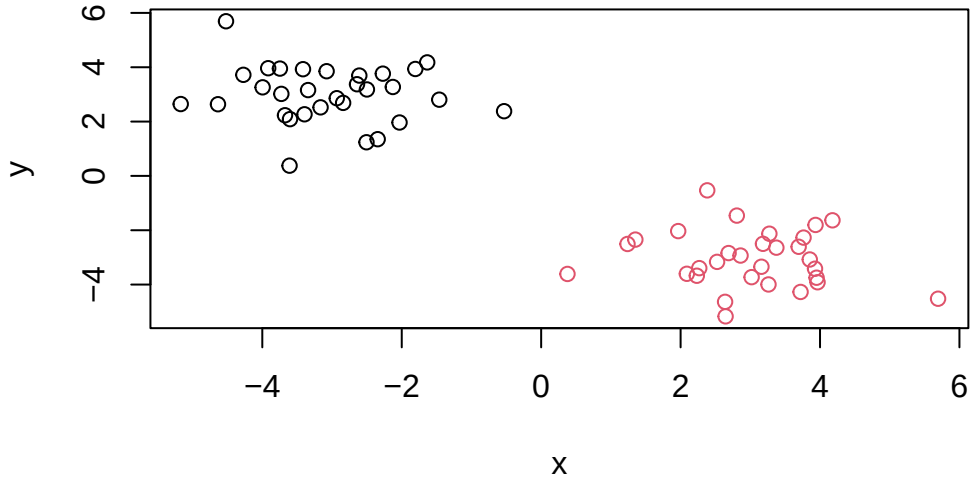


To get my cluster membership vector, I need to "cut" my tree to yield sub-trees or branches with all the members of a given cluster residing on the same cut branch.The function to do this is called `cutree()`

```
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 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
```

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



It is often helpful to use the `k=` argyment to cutree rather than the `h=` height of cutting with `cutree()`, This will cut the tree where it will give you your desired number of clusters.

```
cutree(hc, k=4)
```

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

Principal Component Analysis (PCA)

The base R function for PCA is called `prcomp()` The motivation is to reduce the features dimensionality while only losing a small amount of information. The first principal component (PC) follows a "best fit" through the data points. Principal components are new low dimensional axis (or surfaces) closest to the observations. Can get rid of the old axes, and just use the PC1 and PC2 to describe the data.

Lab

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

Q1.

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

```
[1] 17  5
```

```
## 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
```

```
dim(x)
```

```
[1] 17  4
```

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

```
              England Wales Scotland N.Ireland
Cheese            105   103      103        66
```

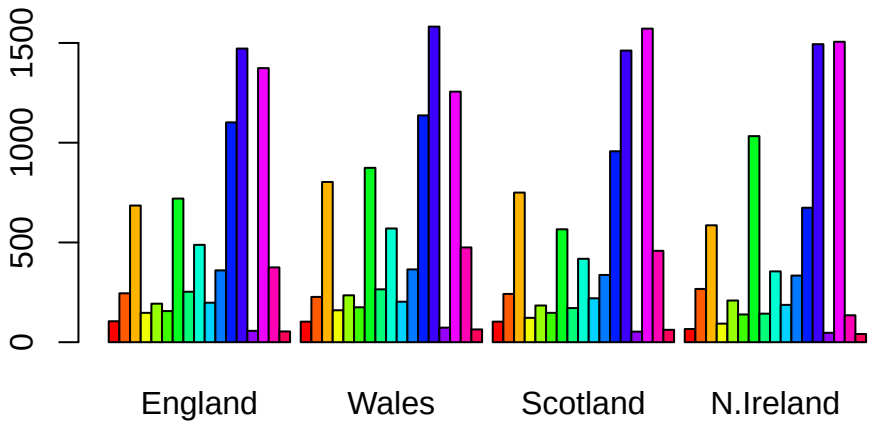
```
Carcass_meat       245   227       242       267
Other_meat         685   803       750       586
Fish               147   160       122        93
Fats_and_oils      193   235       184       209
Sugars             156   175       147       139
```

```
dim(x)
```

```
[1] 17  4
```

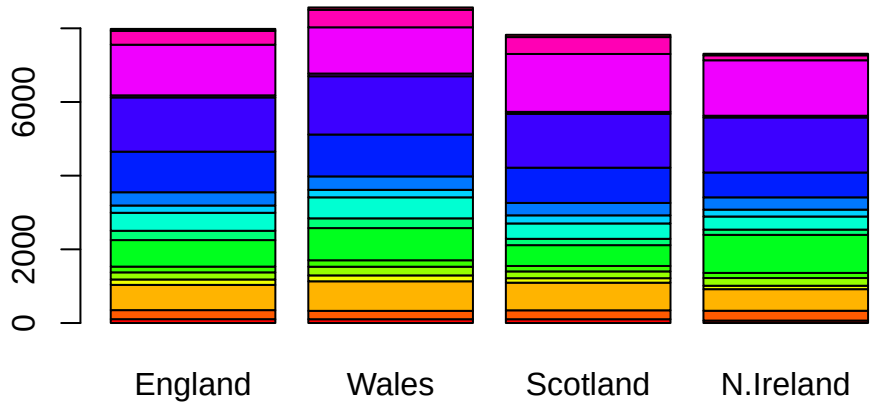
Q2. Using the second method of counting columns is better. If you ru the first method multiple times, the number of columns becomes smaller every time.

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

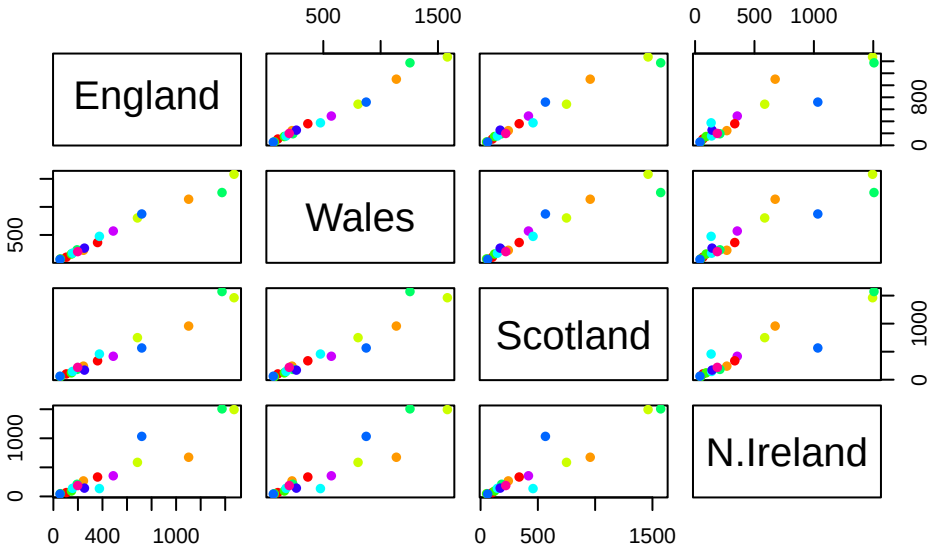


Switching the `beside=` argument to false would result in the bars stacked on top of each other instead of next to each other.

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




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



Q5. The axes are denoted by the countries. If there is a more diagonal line with less scatter, that means that the two countries being compared have similar food consumption patterns.
Q6. People in N. Ireland eat much more fresh potatoes and way less alcoholic drinks compared to the other countries.

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

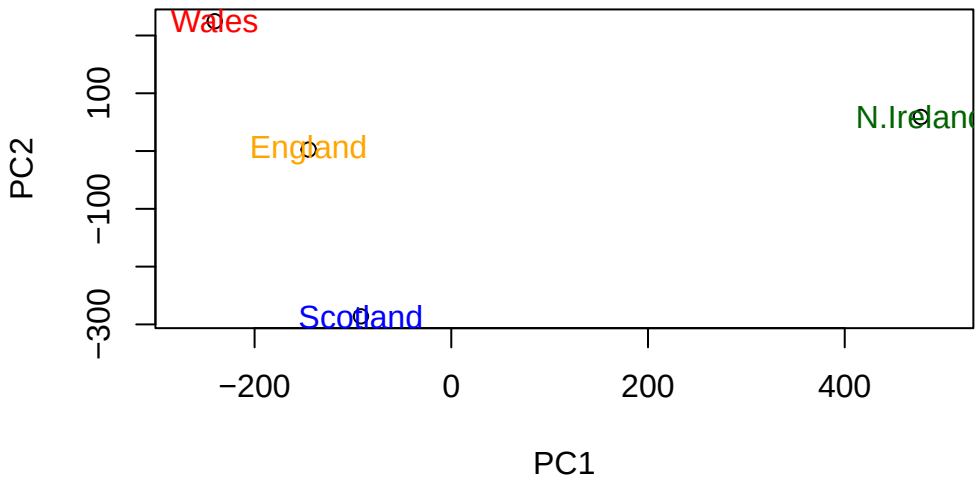
```
Importance of components:
                            PC1      PC2      PC3       PC4
Standard deviation     324.1502 212.7478 73.87622 4.189e-14
Proportion of Variance   0.6744   0.2905  0.03503 0.000e+00
Cumulative Proportion    0.6744   0.9650  1.00000 1.000e+00
```

```
pca$x
```

```
                PC1         PC2         PC3           PC4
England   -144.99315    2.532999 -105.768945  2.842865e-14
Wales     -240.52915  224.646925   56.475555  7.804382e-13
Scotland   -91.86934 -286.081786   44.415495 -9.614462e-13
N.Ireland  477.39164   58.901862    4.877895  1.448078e-13
```

A "PCA plot" (aka Score plot, PC1vsPC2 plot, etc.)

```
# Plot PC1 vs PC2
#Plot the first column of pca (PC1)vs the second column (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), col= c("orange", "red", "blue", "darkgreen",pch=15
```



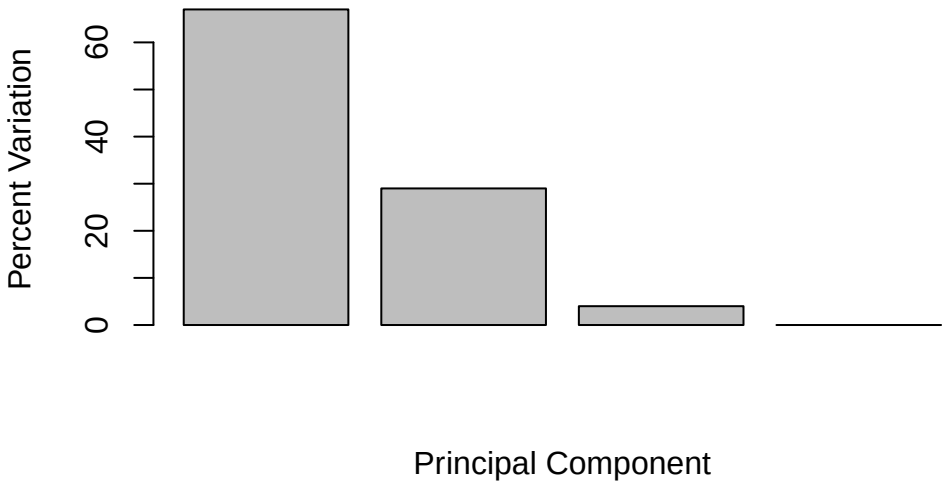

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

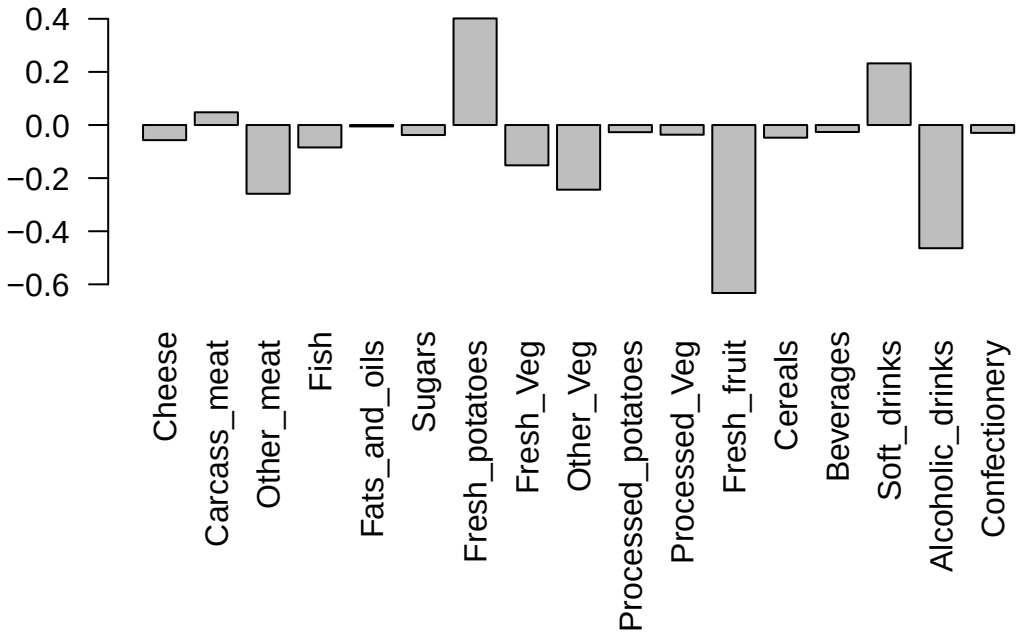
```
## or the second row here...
z <- summary(pca)
z$importance
```

```
                             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
```

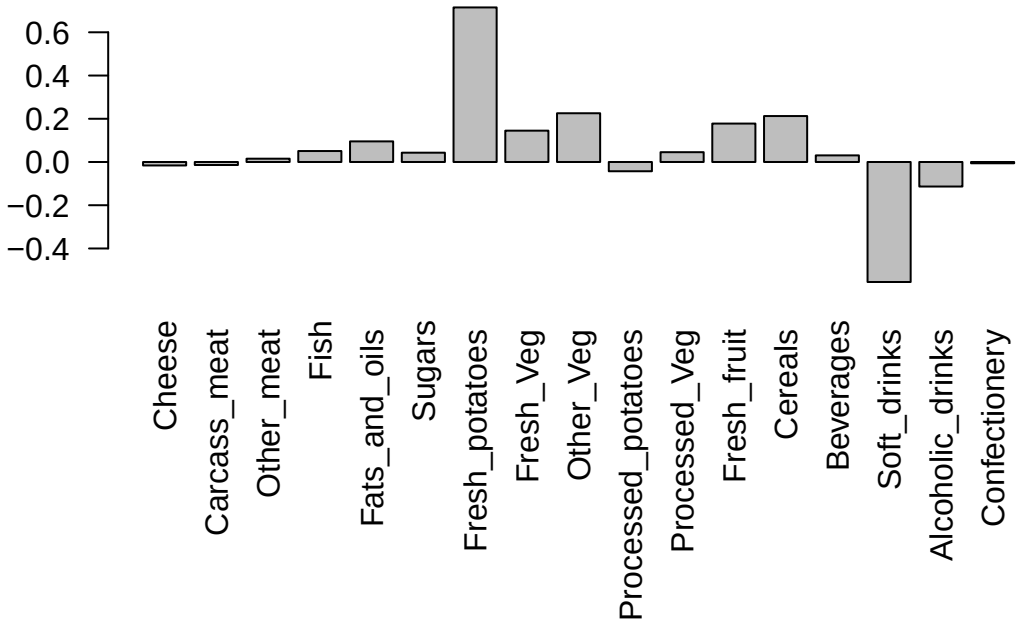
```
barplot(v, xlab="Principal Component", ylab="Percent Variation")
```




```
## 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. PC2 prominently features soft drinks and alcoholic drinks.