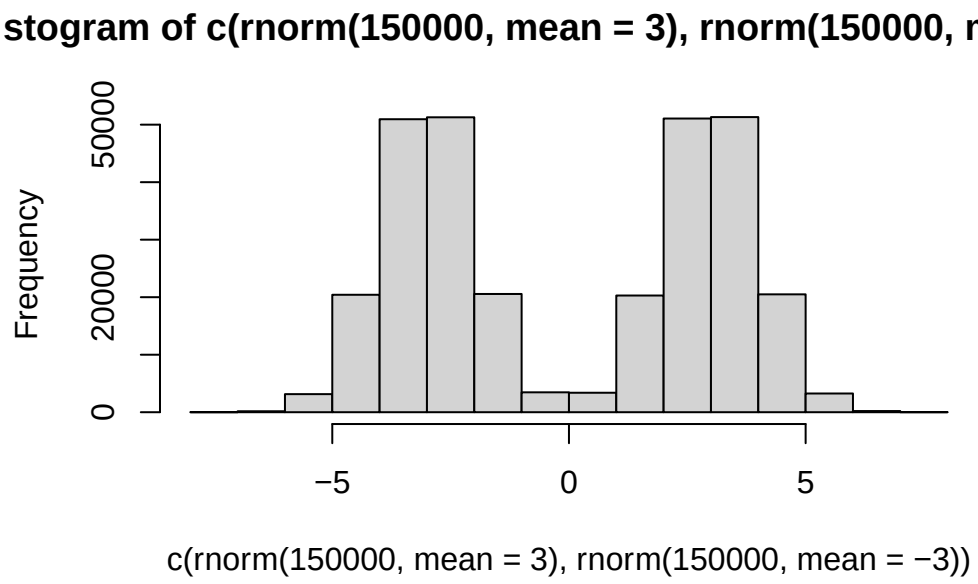
Class 7: Machine learning 1

Anqi Feng (PID:A16243334)

Before we get into clustering methods, let's make some sample datas to cluster where we know what the answer should be.

To help with this, I will use the `rnorm()` function

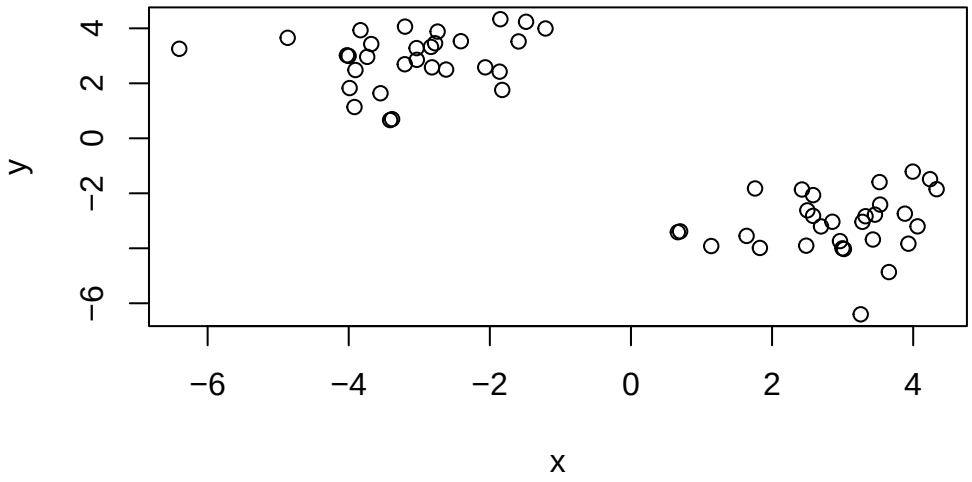
```
hist(c(rnorm(150000, mean=3),rnorm(150000, mean=-3)))
```




```
n=30
x <- c(rnorm(n,mean=3), rnorm(n, mean = -3))
x
```

```
 [1]  3.9939967  0.6967039  1.6388626  1.8277162  3.3244475  3.4578707
 [7]  3.6558198  3.9320549  4.0619903  3.2808866  2.4237542  3.2563784
[13]  1.1376616  2.9624607  3.5231994  3.8817634  2.5002202  0.6640469
[19]  2.4839798  2.5808095  4.2389227  2.8546160  2.6939966  1.7587020
[25]  3.0184049  2.9975158  3.4291162  4.3322635  3.5314916  2.5812406
[31] -2.0653925 -2.4110426 -1.8508250 -3.6809688 -4.0016389 -4.0259660
[37] -1.8243235 -3.2055901 -3.0351580 -1.4868172 -2.8199599 -3.9051042
[43] -3.4135801 -2.6202320 -2.7405310 -1.5925064 -3.7380671 -3.9186942
[49] -6.4016955 -1.8617895 -3.0387282 -3.2021124 -3.8330983 -4.8651943
[55] -2.7768275 -2.8342760 -3.9874013 -3.5501578 -3.3853378 -1.2113610
```

```r
y <-rev(x)
z<- cbind(x,y)
plot(z)
```



##K-means clustering The function in base R, the main function for kmeans clustering is called `kmeans()`. kmeans(x, centers, iter.max = 10, nstart = 1, algorithm = c(…), trace = FALSE) centers: num of clusters or "k"

```r
km<- kmeans(z,centers = 2)
```

```
km$centers
```

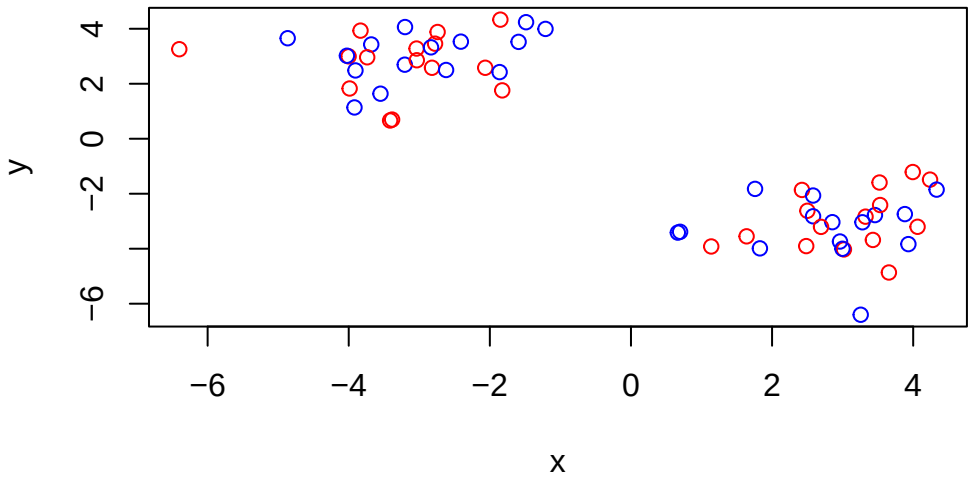
```
          x         y
1  2.890696 -3.109479
2 -3.109479  2.890696
```

Q: print out the cluster membership vector (ie our main answer)

```
km$cluster
```

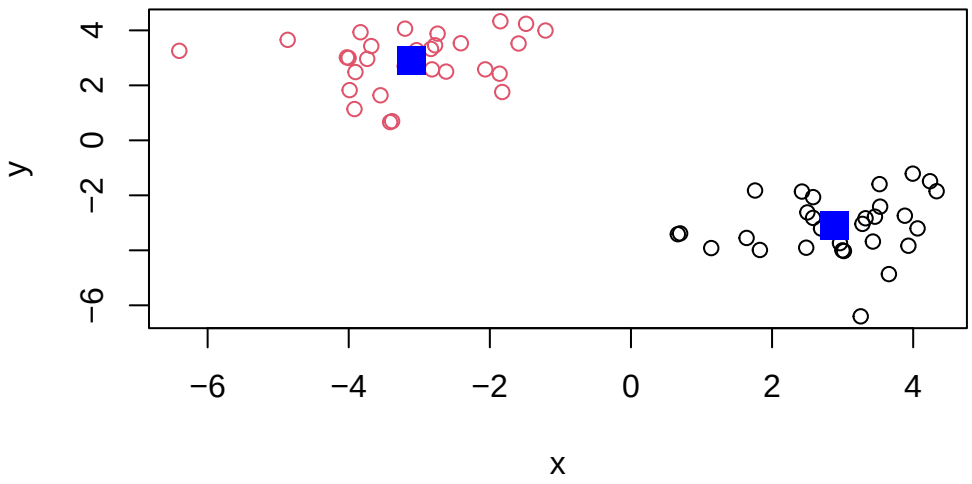
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

```
plot(z, col = c("red", "blue"))
```



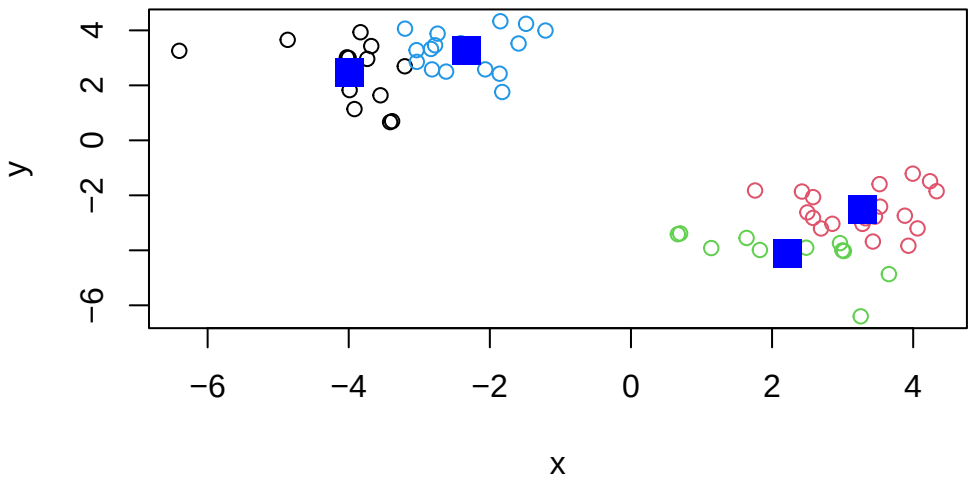
Plot with clustering result and add cluster centers:

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



Q. Can you cluster our data in 'z' into four clusters please?

```
km4 <- kmeans(z,centers = 4)
plot(z, col=km4$cluster)
points(km4$centers, col="blue", pch=15, cex=2)
```



##Hierarchical clustering The main function to do hierarchical clustering in base R is called `hclust()` unlike `kmeans()`I cannot just pass in my data as input I first need a distance matrix from my data.

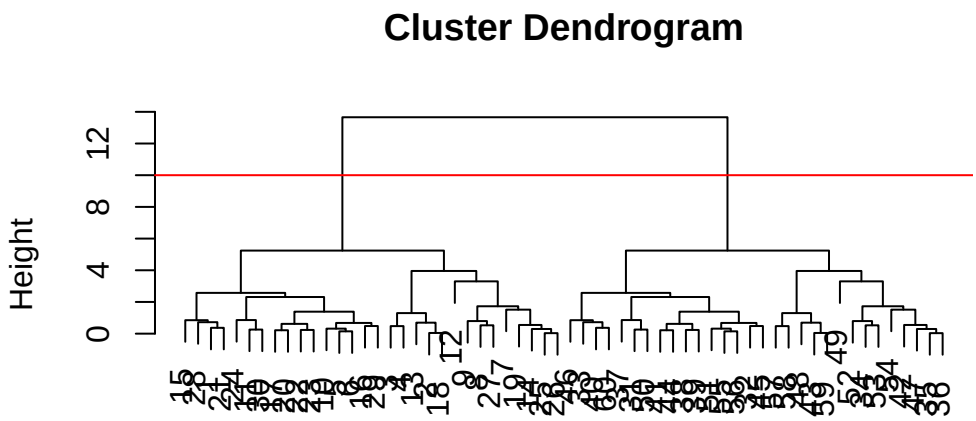
```
d <- dist(z)
hc <- hclust(d)
hc
```

```
Call:
hclust(d = d)

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

There is a specific hclust plot method ...

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



To get my main clustering result(i.e. the membership vector) I can "cut" my tree at a wanted height, to do this I will use the `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
```

##PCA is a well established multivariable statisticall tool to decrease dimentionality ##UK food problem example exercise

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

```
[1] 17  5
```

```
##Preview the first 6 rows
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
```

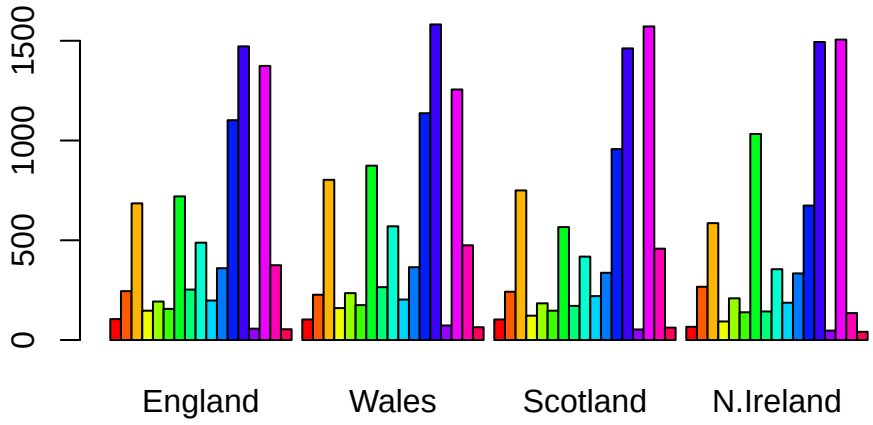
```
dim(x)
```

```
[1] 17  4
```

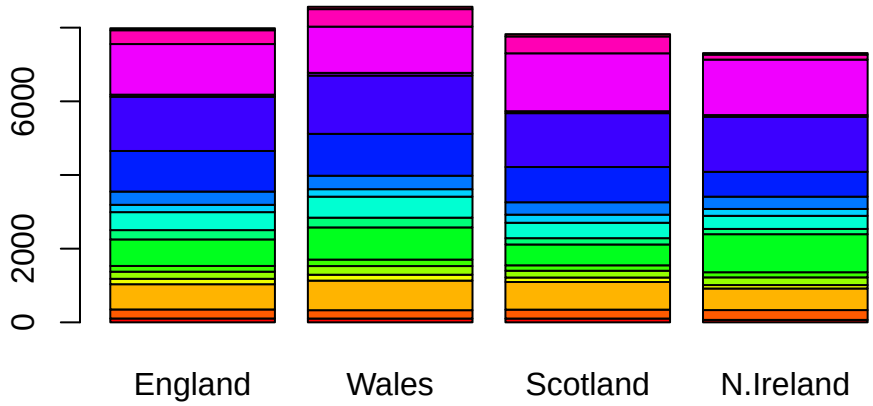
```
x <- read.csv(url, row.names=1)
head(x)
```

```
               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
```

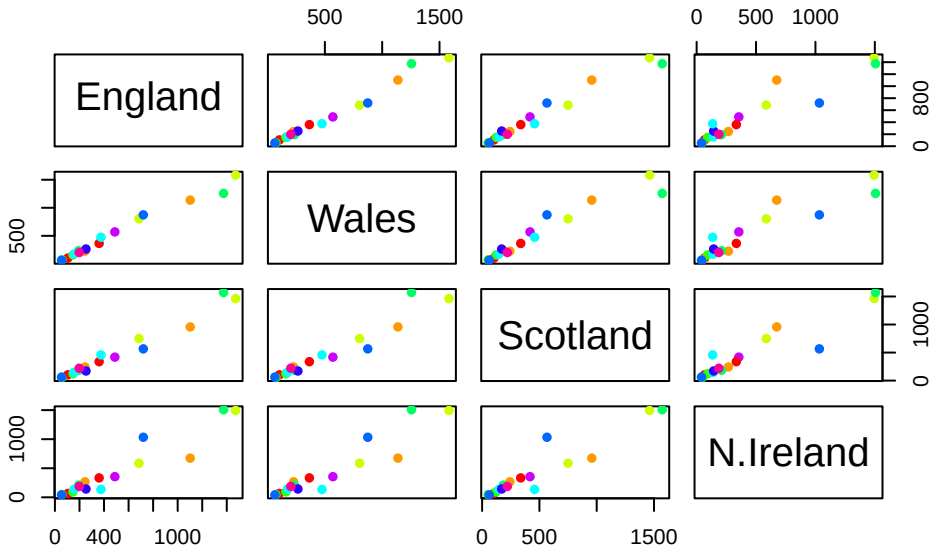
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```




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



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



##PCA to the rescue the main function of PCA is called `prcomp()` Note that I need to take the transpose this particular data as that is what the `prcomp()` help page was asking for

```
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
```

Let's see what is inside our result object pca that we just calculated:

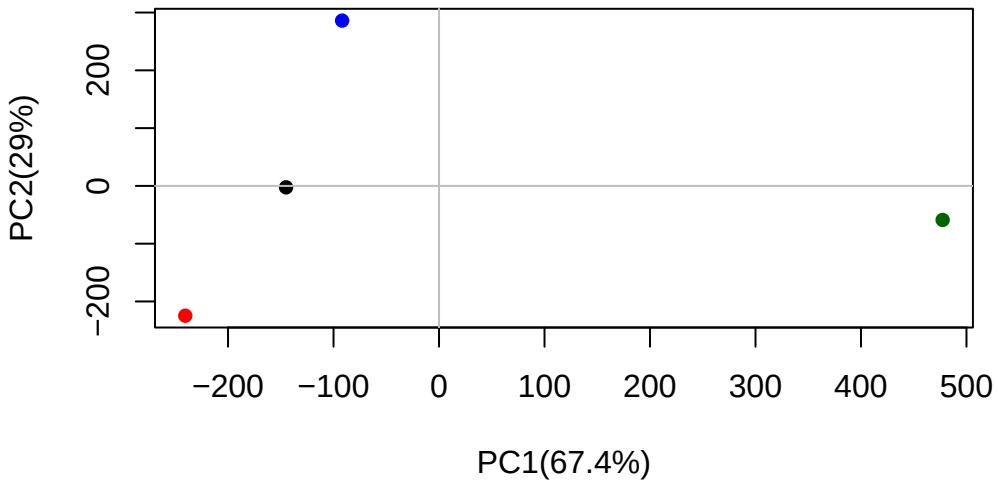
```
attributes(pca)
```

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

```
$class
[1] "prcomp"
```

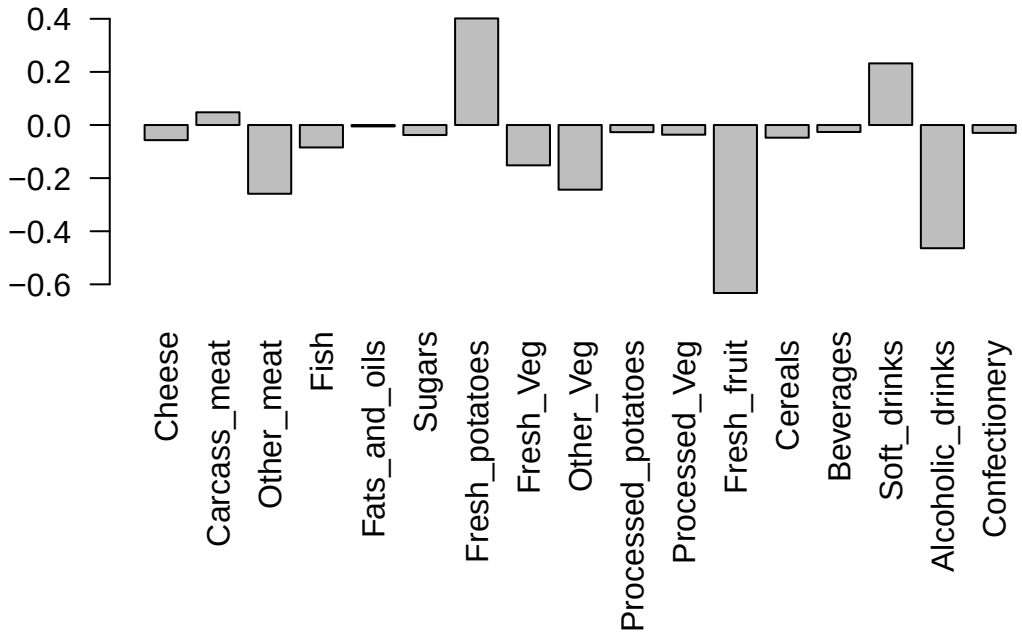
To make a results figure of the “PC plot” or “score plot” or “coordination plot” or “PC1 vs PC2 plot”.

```
plot(pca$x[,1], pca$x[,2], col=c("black", "red", "blue", "darkgreen"),pch = 16, xlab="PC1(67
abline(h=0, col="gray")
abline(v=0, col="gray")
```



##Variable Loadings Can give us insights on original variables(in tihs case the foods) contribute to >90% of the variants

```
## 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 )
```



Fresh potatos and soft drinks are more in scotland, and less fresh fruit and alcoholic drinks compared to others