Class07: Machine Learning1

Juan Manuel Tzompantzi De Ita (PID:A69034758)

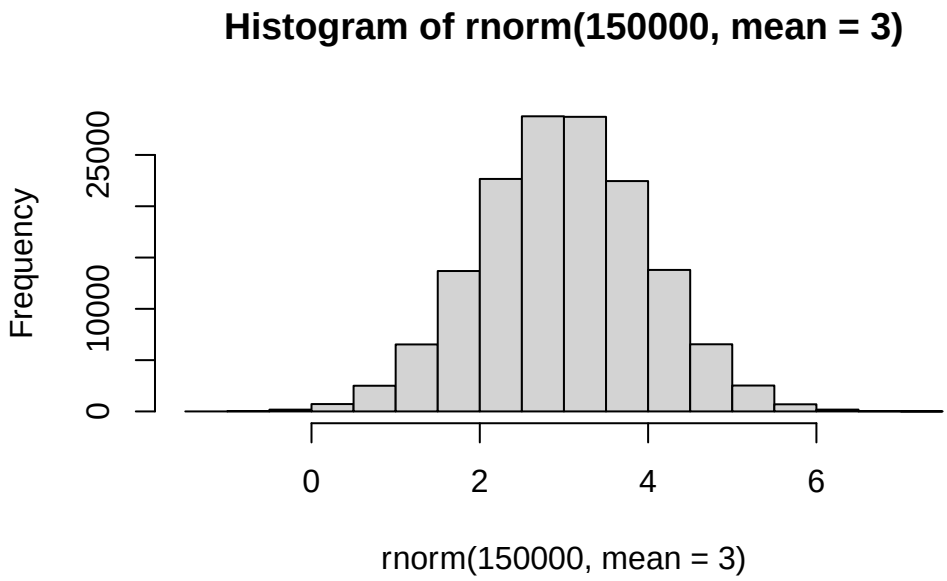
Before we get into clustering methods let's make some sample data to cluster where we know what the answer should be…

To help with this we will use `rnom()` fucntion.

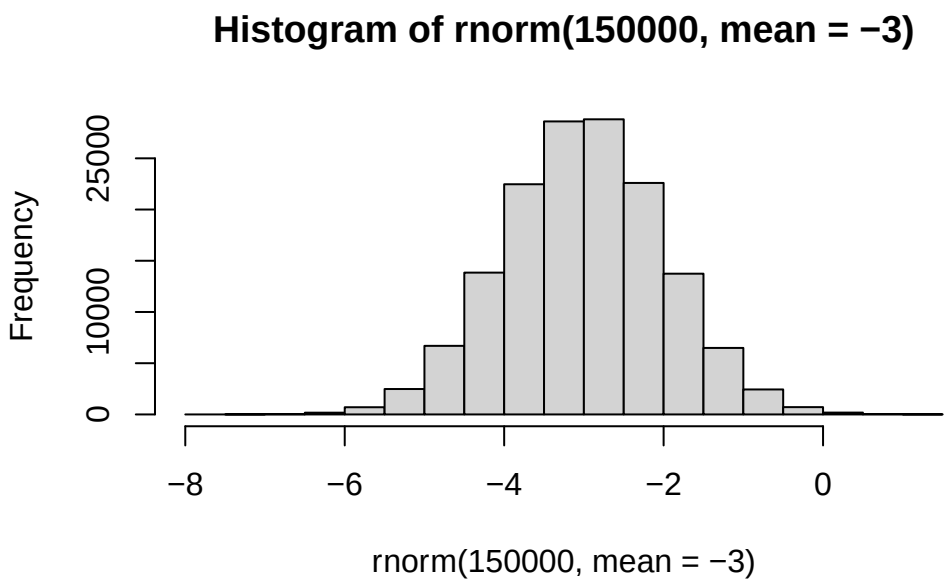
```
rnorm(10)
```

```
 [1] -1.07390869 -1.56904483  0.78451078  0.99477191  0.26888469  0.45538466
 [7]  0.16288534  2.45168659 -0.06071673 -0.45432210
```

```
p <- hist( rnorm(150000, mean=3))
```

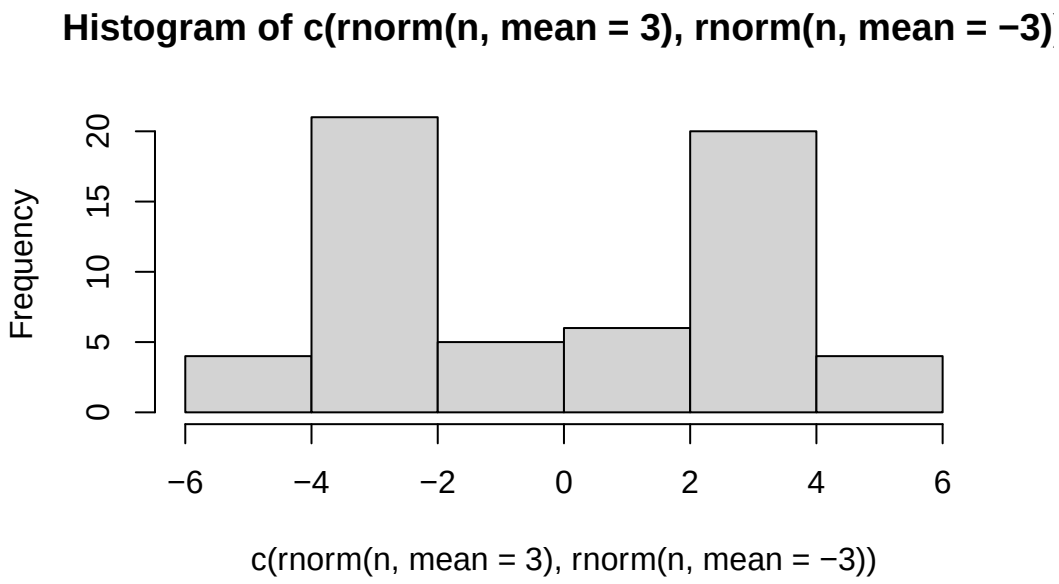


```
hist( rnorm(150000, mean=-3))
```



To plot both data sets, we can vectorized the data.

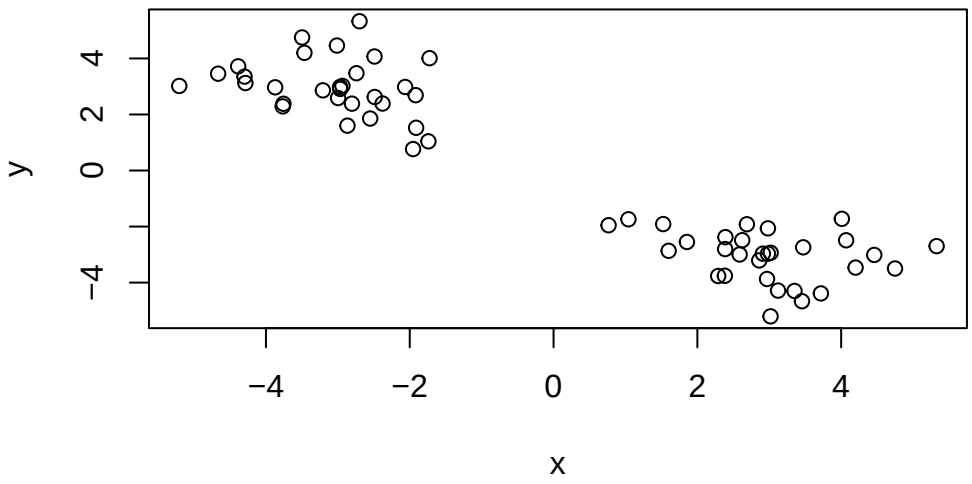
```
n = 30
hist( c( rnorm(n, mean=3), rnorm(n, mean=-3) ))
```



Let's say we wanna create a multidimentional data set:

We can use the same distribution but apply different rules for the different axes.

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


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



K-means clustering

The function in base R for k-means clustering is called `kmeans()`

```
?kmeans()
```

```
starting httpd help server ... done
```

```
kmeans(z, centers=2)
```

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

Cluster means:
          x         y
1  2.960265 -3.053874
2 -3.053874  2.960265

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 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] 55.70248 55.70248
 (between_SS / total_SS =  90.7 %)

Available components:

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

If we wanna know that the components are we can rewrite it:

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

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

Cluster means:
          x         y
1 -3.053874  2.960265
2  2.960265 -3.053874

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] 55.70248 55.70248
 (between_SS / total_SS =  90.7 %)

Available components:

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

And then explore the components with:

```
km$centers
```

```
          x         y
1 -3.053874  2.960265
2  2.960265 -3.053874
```

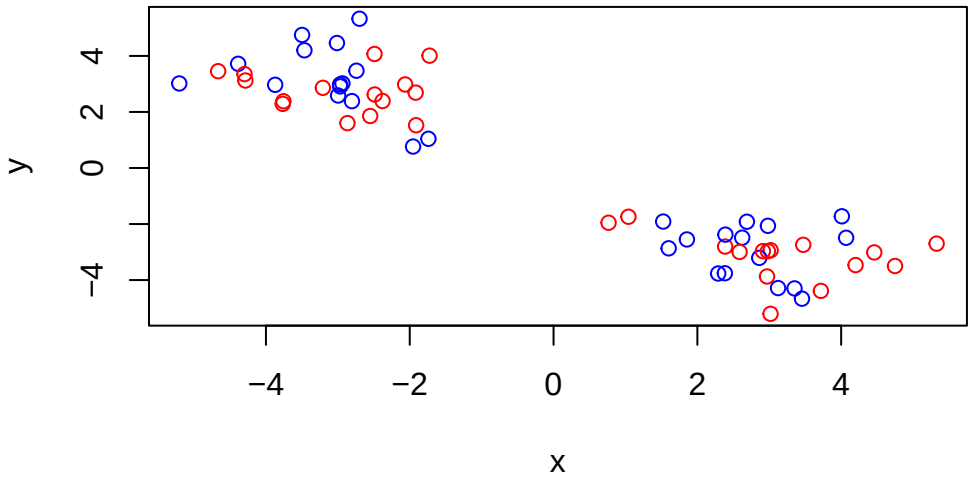
Q. Print out the cluster membership vector (i.e. our main answer)

```
km$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
```

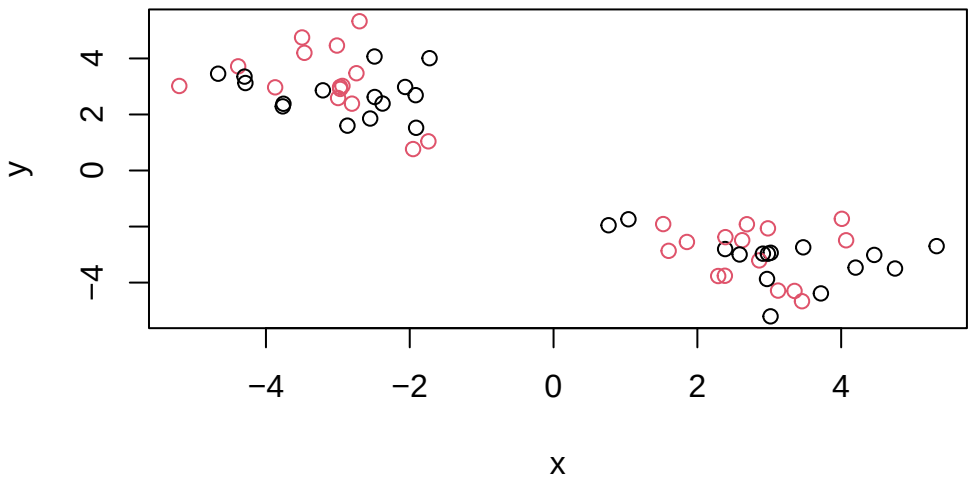
If we wanna color out plots, this command will give us a plot that iterates between colors...

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



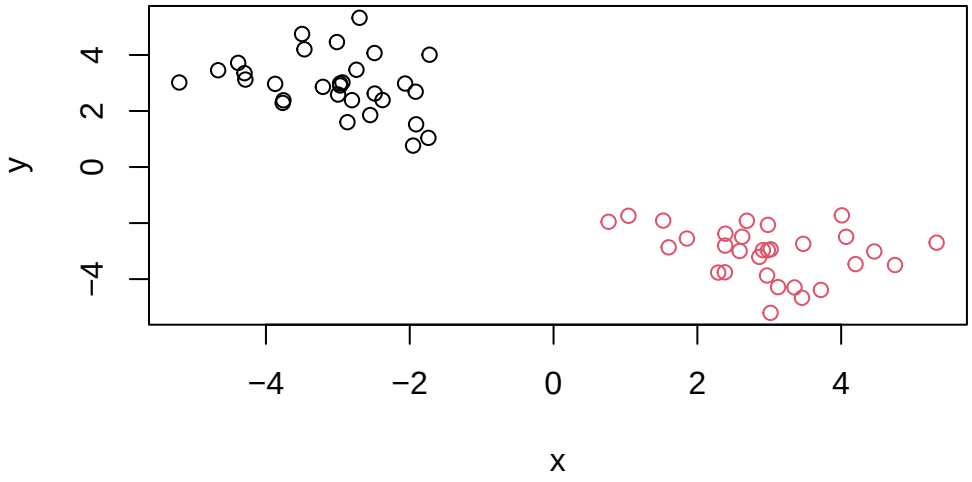
Same with this one:

```
plot(z, col=c(1,2))
```



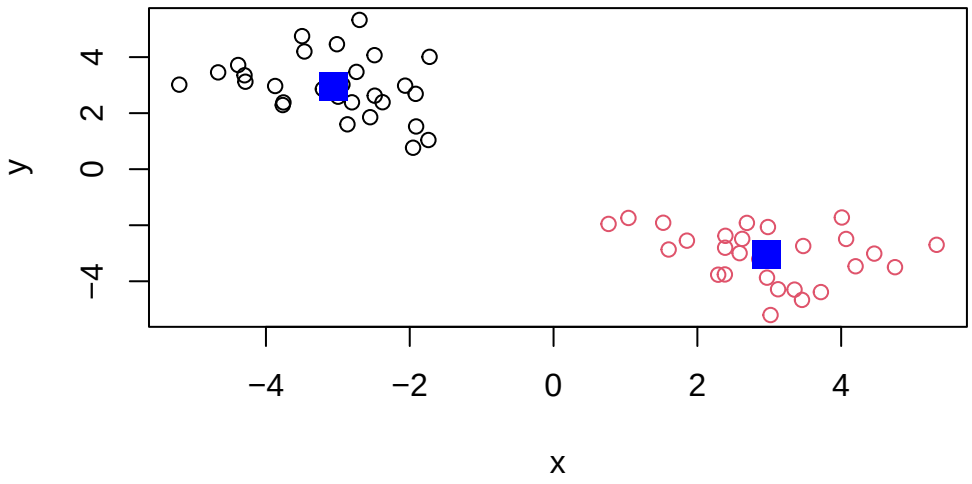
We need to do this:

```
plot(z, col=km$cluster)
```



When, plot with clustering results and add cluster centers:

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



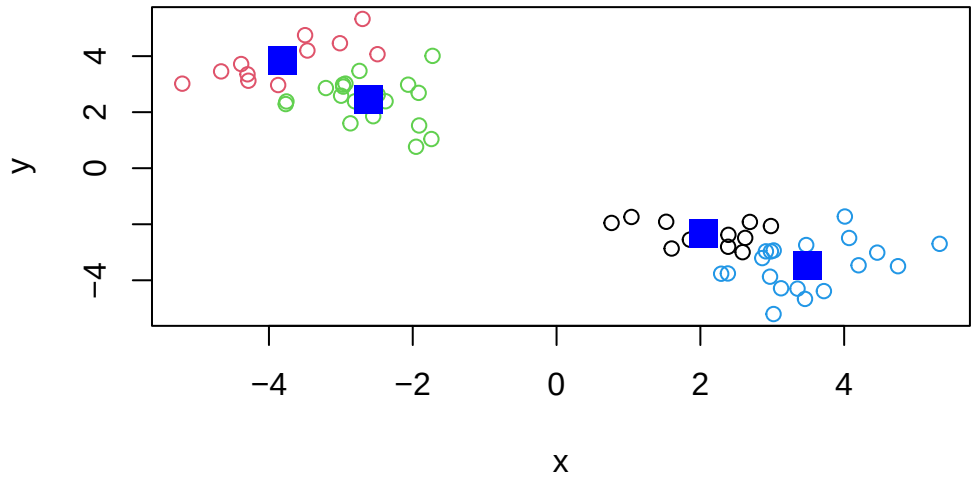
Q. Can you cluster our data in z into four clusters?

```
km4 <- kmeans(z, centers=4)
km4
```

```
K-means clustering with 4 clusters of sizes 11, 11, 19, 19

Cluster means:
          x         y
1  2.040304 -2.333691
2 -3.806932  3.858043
3 -2.617893  2.440498
4  3.492873 -3.470822

Clustering vector:
 [1] 4 4 4 1 4 1 4 4 1 4 4 4 4 1 4 4 1 4 4 1 4 1 4 1 4 4 1 4 1 1 3 3 2 3 3 3 3 2
[39] 3 3 3 2 3 3 3 2 3 3 2 2 2 3 2 2 3 2 3 2 3 3
```

```
Within cluster sum of squares by cluster:
[1]  7.241558 13.260971 18.592885 24.753199
 (between_SS / total_SS =  94.7 %)

Available components:

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

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



##Hierarchical Clustering

The main function for hierarchical clustering `hclust()`

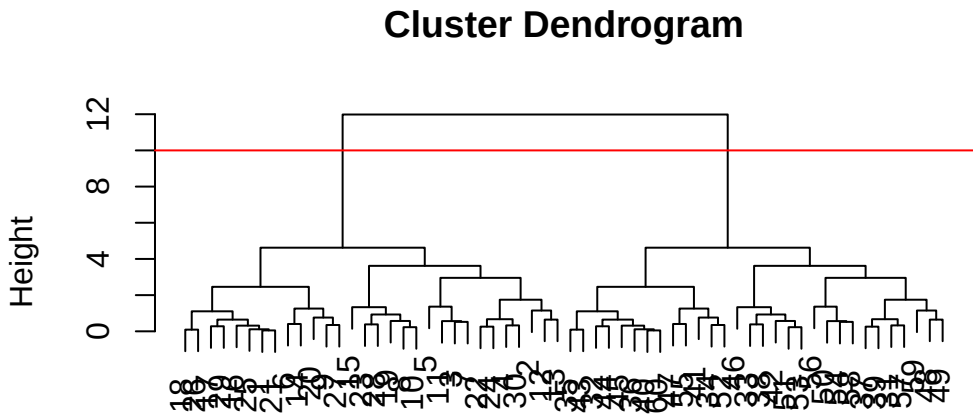
Unlike `kmeans()`, I cannot just pass in my data as input. I first need a distance matrix from my data.

```
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 for this method.

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

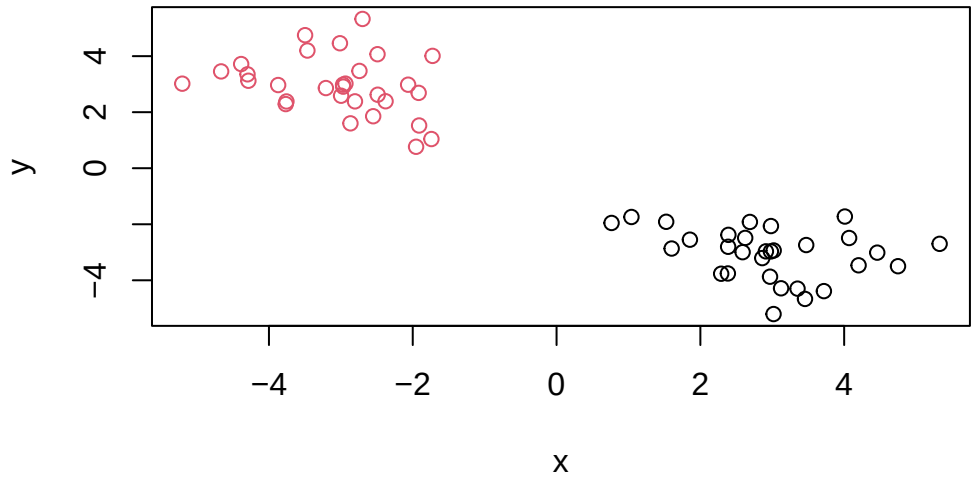


To get my main clustering result (i.e. the membership vector), I can "cut" my tree at a given height. To do this I will use the: `cutree()`:

```
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
```

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



##Principal Component Analysis PCA prjects the features onto the principal components. The motivation is to decude the futures dimensionality while only losing a small amount of information.

Principal component analysis (PCA) is a well established "multivariate statistical technique" used to reduce the dimensionality of a complex data set to a more manageable number (typically 2D or 3D). This method is particularly useful for highlighting strong paterns and relationships in large datasets (i.e. revealing major similarities and diferences) that are otherwise hard to visualize.

Hands-on practice!

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names = 1) #eliminating the first column

x
```

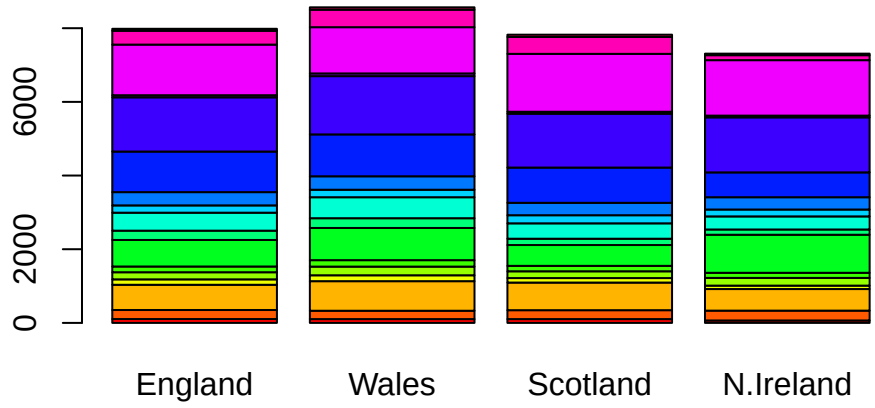
```
                England Wales Scotland N.Ireland
Cheese              105   103      103        66
Carcass_meat        245   227      242       267
Other_meat          685   803      750       586
```

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

```
dim(x)
```

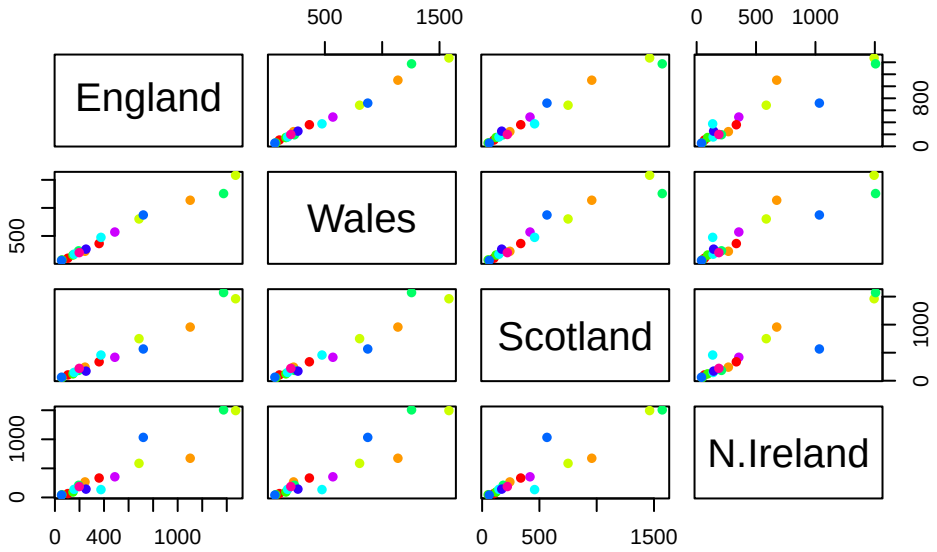
```
[1] 17  4
```

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



Trying to understand the main differences with a lot of plots...

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



As we can see, spotting main differences is hard even for only 4 countries and 17 dimensions.. We are gonna be using the PCA. The main function of R base for PCA is `prcomp()`

```
pca <- prcomp( t(x) ) #Changing order of columns and rows.
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
```

For the summary above, we can tell that PC1, PC2 and PC3 are sufficient enough to explain our data since they contribute to almost all the variance of our 17 dimensions.

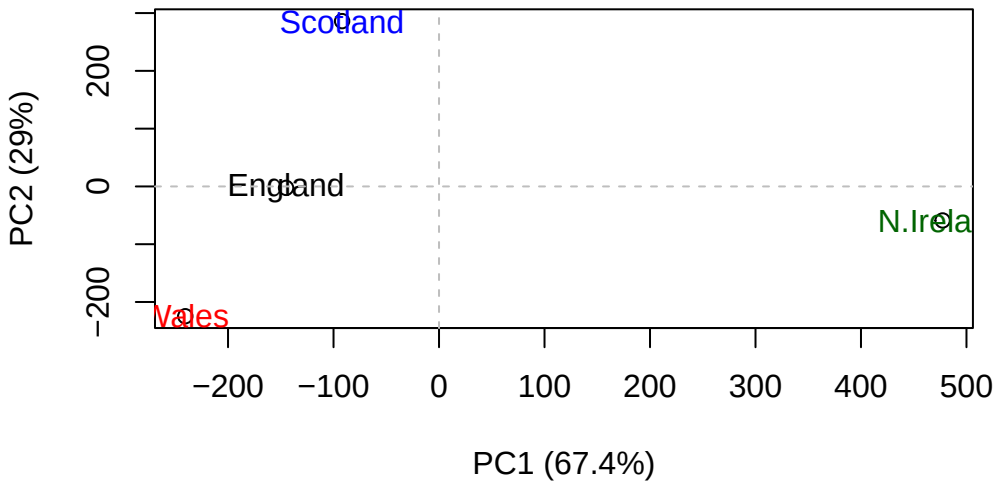
To see what is inside of our result object `pca` that we just created

```
attributes(pca)
```

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

$class
[1] "prcomp"
```

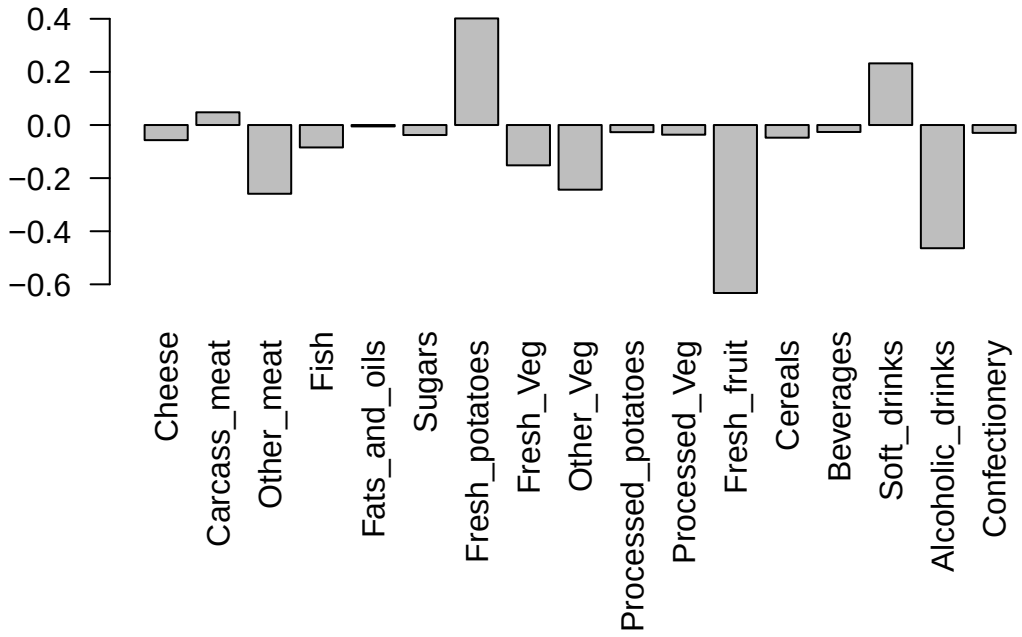
To make the main result figure, called a "PC plot" (or "score plot", "ordination plot" or "PC1 vs PC2 plot")

```
plot(pca$x[,1], pca$x[,2], xlab="PC1 (67.4%)", ylab="PC2 (29%)")
text(pca$x[,1], pca$x[,2], colnames(x), col=c("black","red","blue","darkgreen"))
abline(v=0, col="grey", lty=2)
abline(h=0, col="grey", lty=2)
```



#Variable loading plots Digging deeper into the data. Since we see that PC1 component gives most of the variance, we can plot this component and see what the data looks like here.

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


0.4
0.2
0.0
−0.2
−0.4
−0.6
Cheese
Carcass_meat
Other_meat
Fish
Fats_and_oils
Sugars
Fresh_potatoes
Fresh_Veg
Other_Veg
Processed_potatoes
Processed_Veg
Fresh_fruit
Cereals
Beverages
Soft_drinks
Alcoholic_drinks
Confectionery