Class 7: Machine Learning I

Samson A16867000

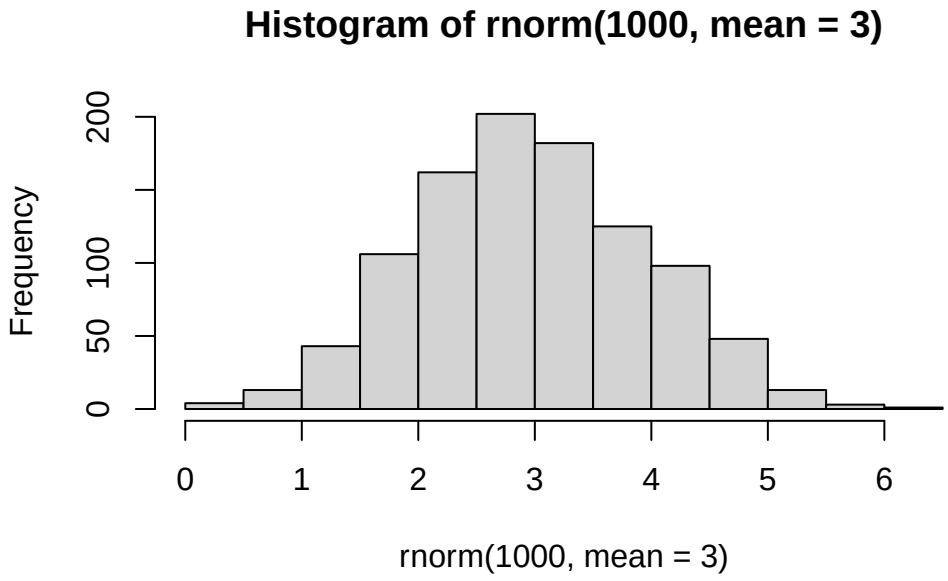
Today we are going to learn how to apply different machine learning methods, beiginning with clustering

The goal here is to find groups/clusters in your input data

```
rnorm(10)
```

```
 [1] -0.7598441 -0.8805147 -0.8584824  1.1908325  0.9315317 -0.5826394
 [7]  0.9714952  0.4369515 -1.2793728  0.8065526
```

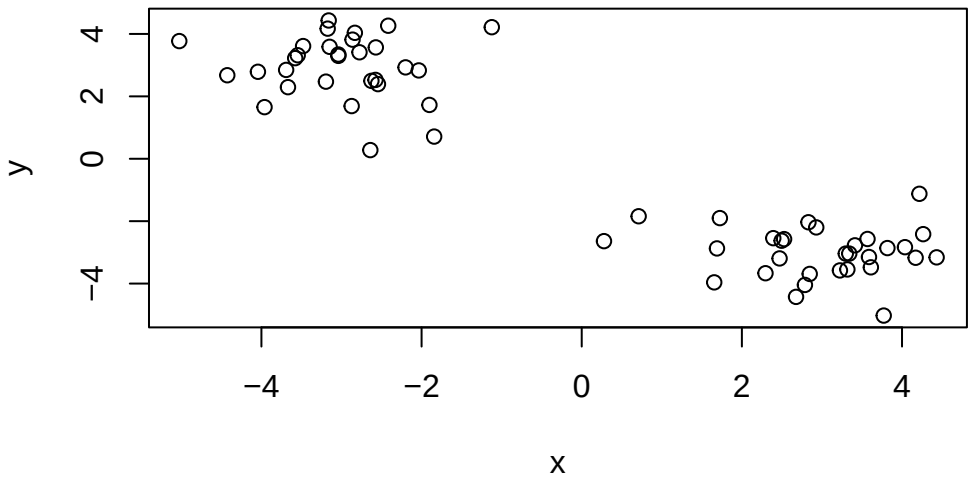
```
hist(rnorm(1000, mean =3))
```



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

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

```
             x        y
[1,] -3.961097 1.654731
[2,] -3.159755 4.432146
[3,] -5.025687 3.771401
[4,] -4.426220 2.676595
[5,] -1.901901 1.725192
[6,] -2.034775 2.831242
```

```
plot(z)
```



Use the kmeans() function setting k to 2 and nstart=20 Inspect/print the results

. Q. How many points are in each cluster? . Q. What 'component' of your result object details - cluster size? - cluster assignment/membership? - cluster center? . Q. Plot x colored by the kmeans cluster assignment and add cluster centers as blue points

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

```r
attributes(km)
```

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

$class
[1] "kmeans"
```

cluster size

```r
km$size
```

```
[1] 30 30
```

cluster membership?

```r
km$cluster
```

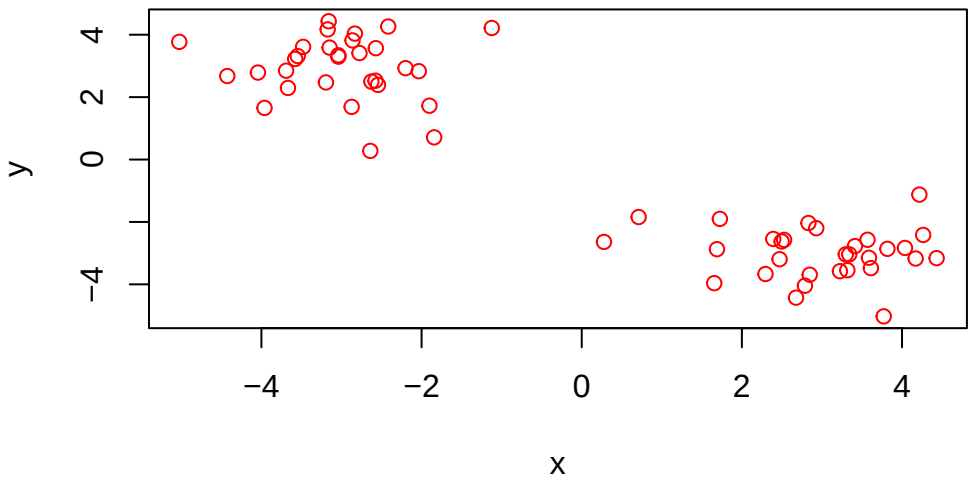
```
 [1] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1
[39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

cluster center?

```r
km$centers
```

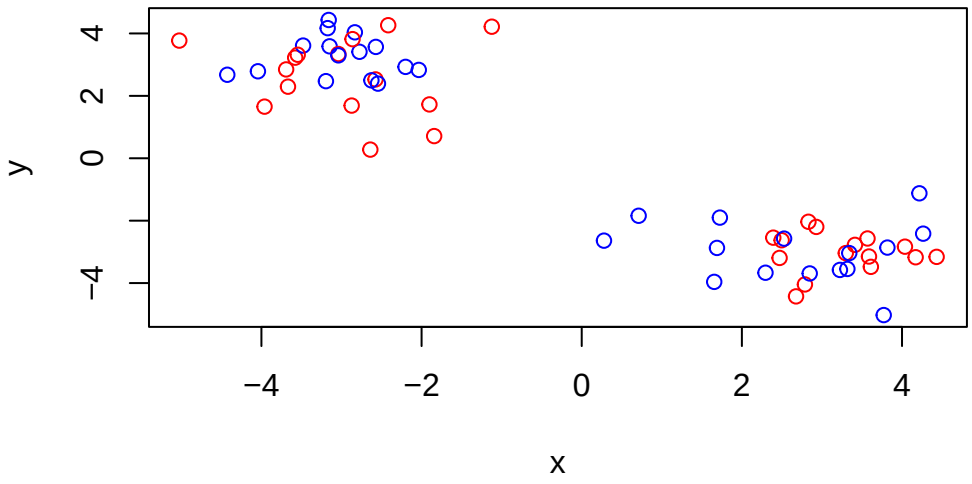
```
          x         y
1  2.945962 -2.999783
2 -2.999783  2.945962
```

```r
plot(z, col = "red")
```

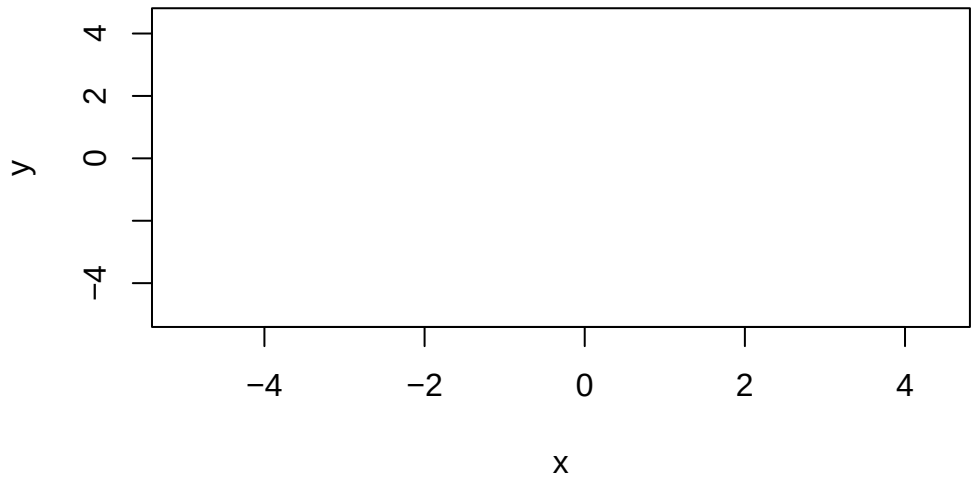


R will recycle the shorter color vector to be the same length as the longer (number of data points) in z

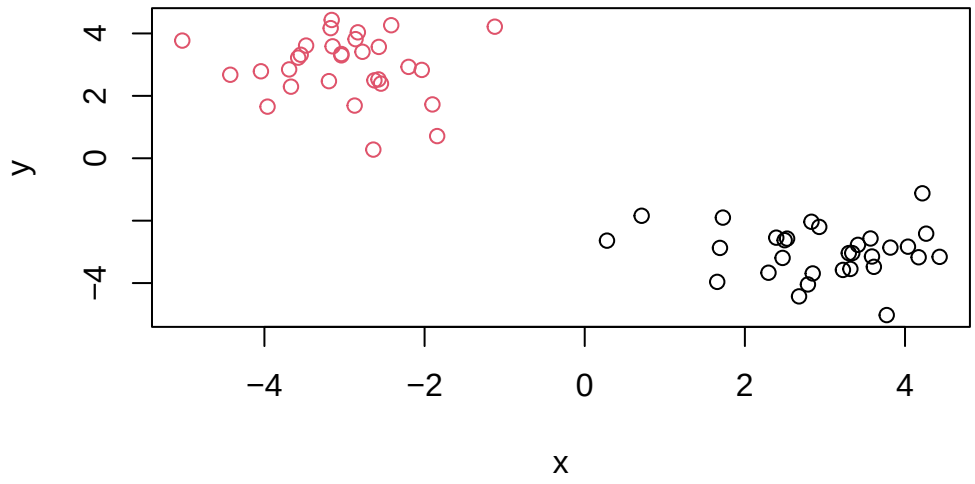
```
plot(z, col = c("red", "blue"))
```



```r
plot(z, col = c())
```




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

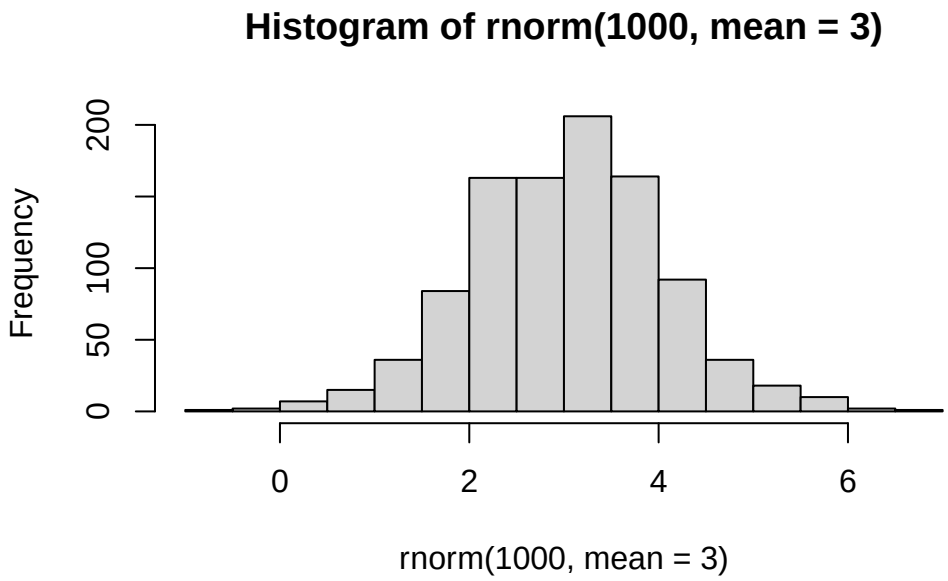


We can use the `points()` function to add new points to an existing plot... like th ecluter centers

```
rnorm(10)
```

```
 [1] -0.86458990  1.45633462  0.72431616 -0.04914211  1.41123885  1.09147107
 [7]  1.75483845  0.95316892  0.27676693 -1.04954082
```

```
hist(rnorm(1000, mean =3))
```

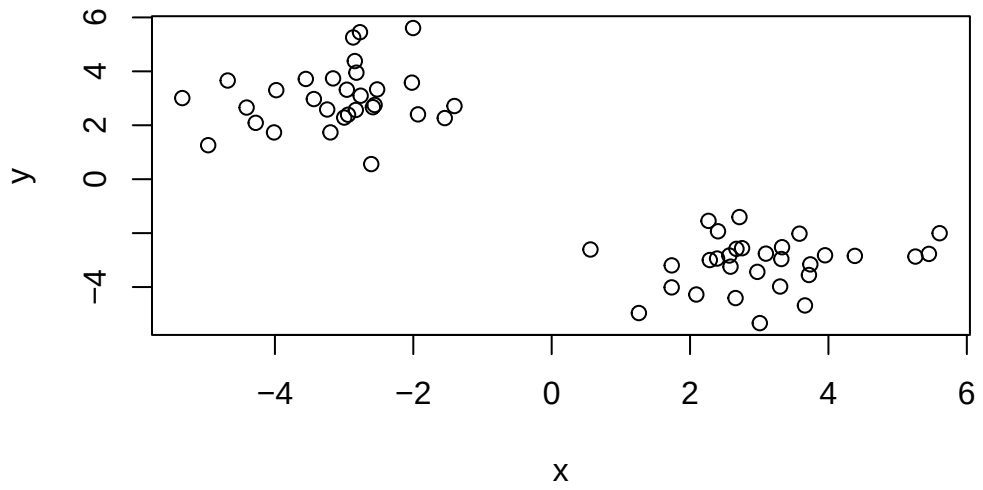



```
n <- 30
x <- c(rnorm(n,-3), rnorm(n, +3 ))
y <- rev(x)

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

```
             x        y
[1,] -2.020228 3.581314
[2,] -2.770879 5.451884
[3,] -3.553626 3.718014
```

```
[4,] -4.409034 2.657305
[5,] -3.244393 2.585127
[6,] -3.980694 3.301810
```

```
plot(z)
```

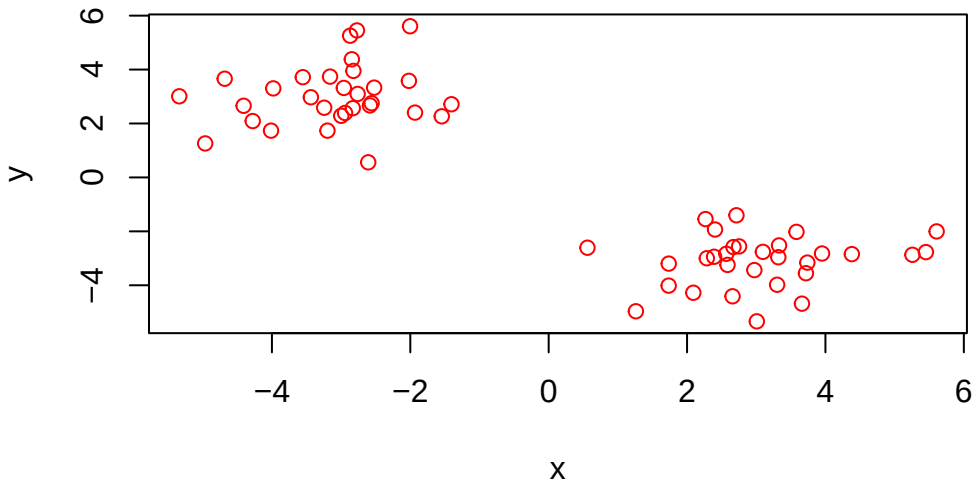



```
km <- kmeans(z, centers = 2)
```

```
attributes(km)
```

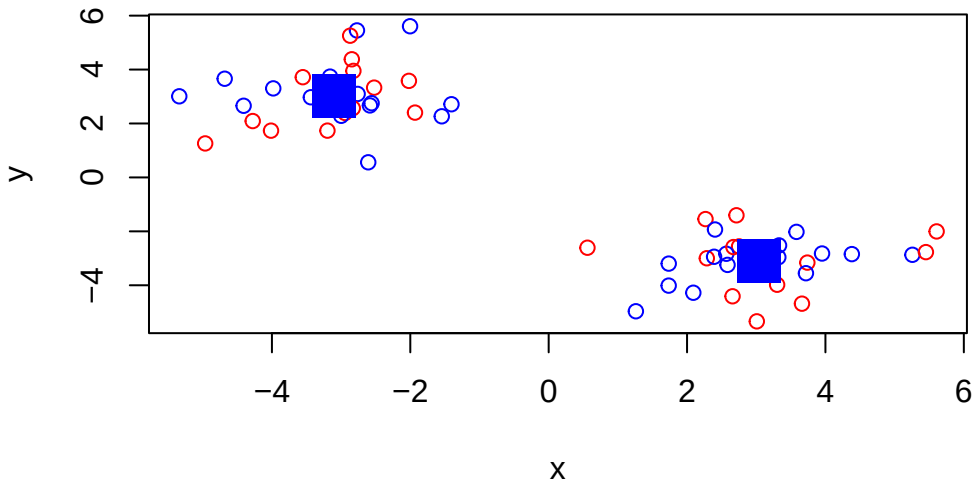
```
$names
[1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
[6] "betweenss"    "size"         "iter"         "ifault"

$class
[1] "kmeans"
```

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



R will recycle the shorter color vector to be the same length as the longer (number of data points) in z

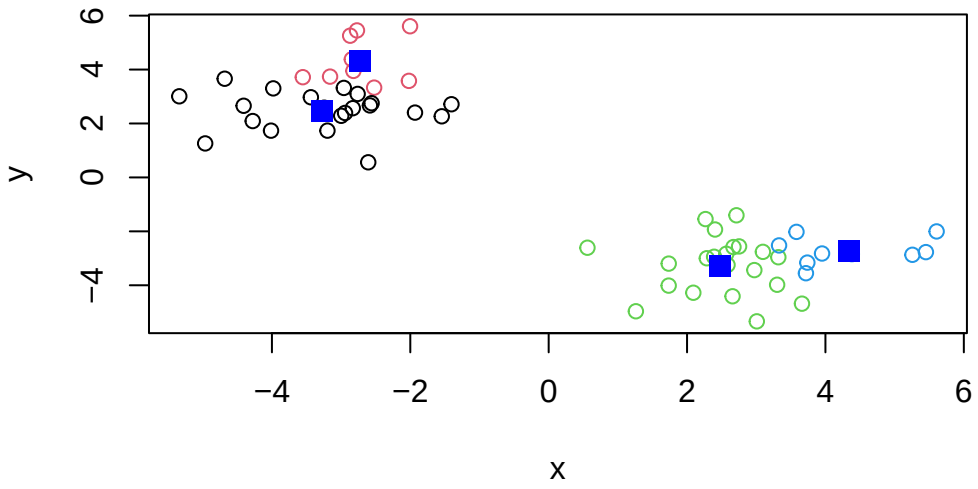
```
plot(z, col = c("red", "blue"))
points(km$centers, col = "blue", pch = 15, cex=3)
```



We can use the `points()` function to add new points to an existing plot... like th ecluter centers

. Q. Can you run kmeans and ask for 4 clusters please and plot the results like we have done above?

```
km4 <- kmeans(z, centers = 4)
plot(z, col = km4$cluster)
points(km4$centers, col = "blue", pch = 15, cex=1.5)
```



Hierarchical Ckustering

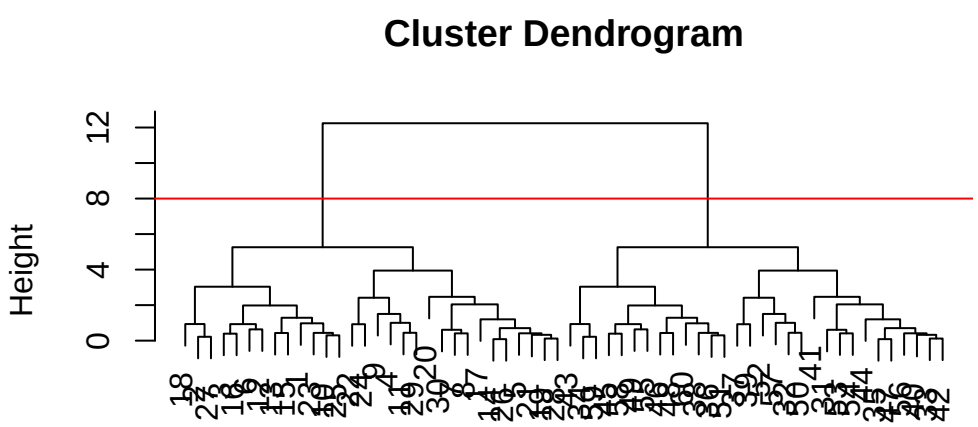
Let's take our same made-up data `z` and see how hclust works.

First we need a distance matrix of our data to be clustered.

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

```
Call:
hclust(d = d)

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

```
plot(hc)
abline(h=8, col = "red")
```



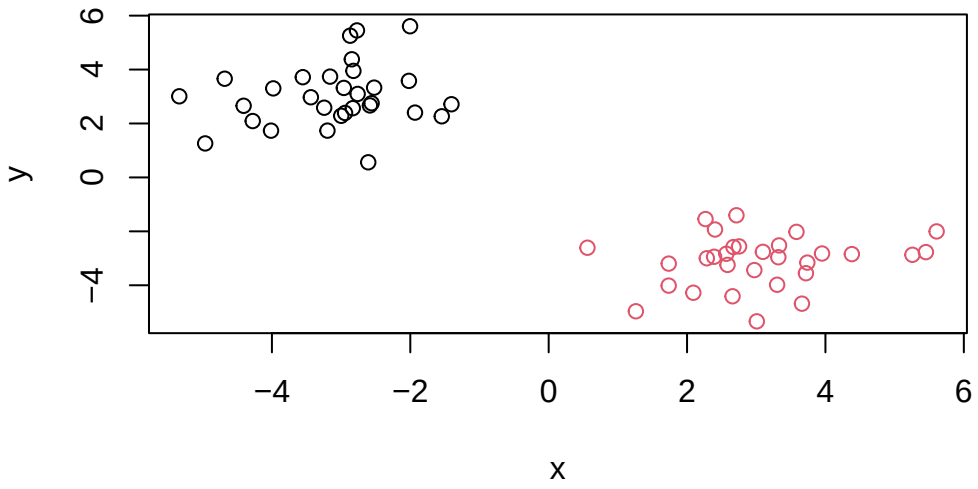
I can get my cluster memebership cluster vector by "clutting the tree" with the `cutree()` function like so:

```
grps <- cutree(hc, h=8)
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 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
```

Can you plt `z` again colored by out hcluster results:

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



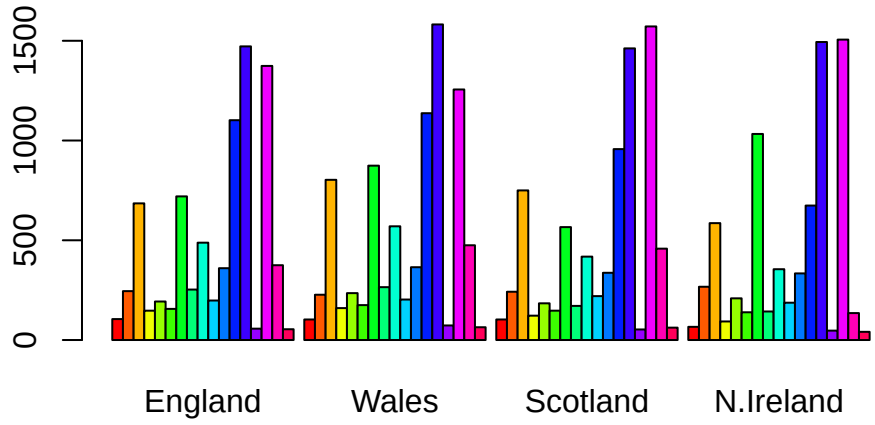
PCA of UK food data

Read data from the UK on food consumption in different parts of the UK

```
url <- "https://tinyurl.com/UK-foods"
x <- read.csv(url, row.names = 1)
head(x)
```

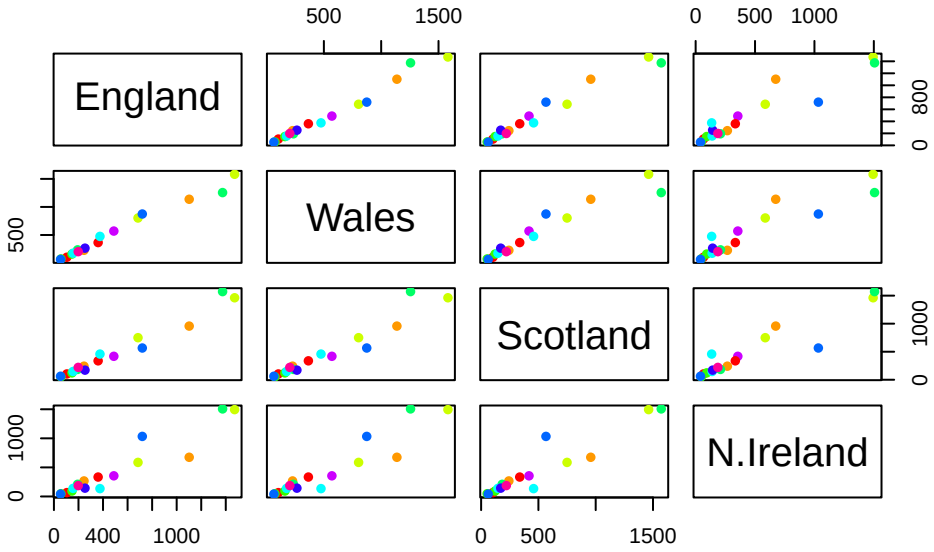
```
              England Wales Scotland N.Ireland
Cheese            105   103      103        66
Carcass_meat      245   227      242       267
Other_meat        685   803      750       586
Fish              147   160      122        93
Fats_and_oils     193   235      184       209
Sugars            156   175      147       139
```

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



so-called "Pairs" plot can be useful for small datasets like

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



It's hard to see structure and trends in even this small dataset how will we wever do this when we have big datasets with 1,000s or 10s of thousands of things we are measuring...

PCA to the rescue

Let's see how PCA deals with this dataset. So the main function in base R to do PCA is called `prcomp()`

```
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
```

Let's see what is inside this PCA object that we created from running `prcomp()`

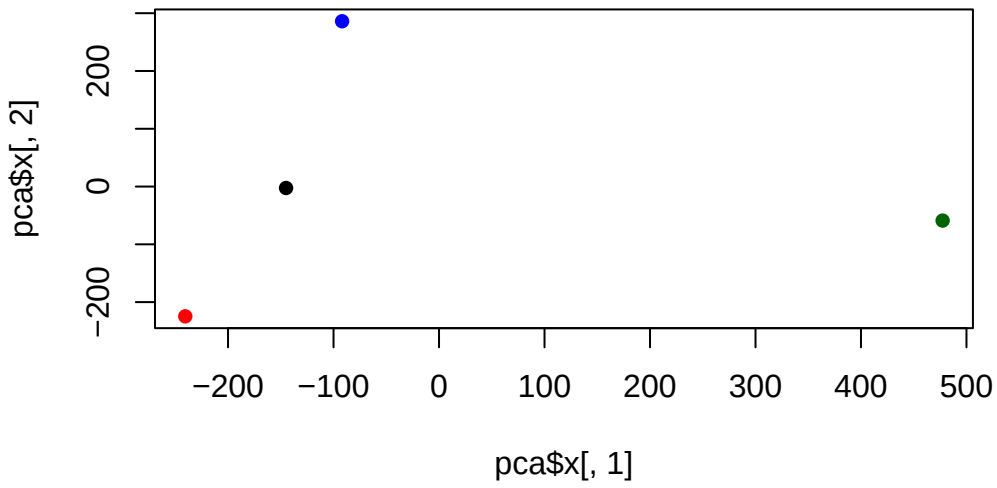
```
attributes(pca)
```

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

$class
[1] "prcomp"
```

```
pca$x
```

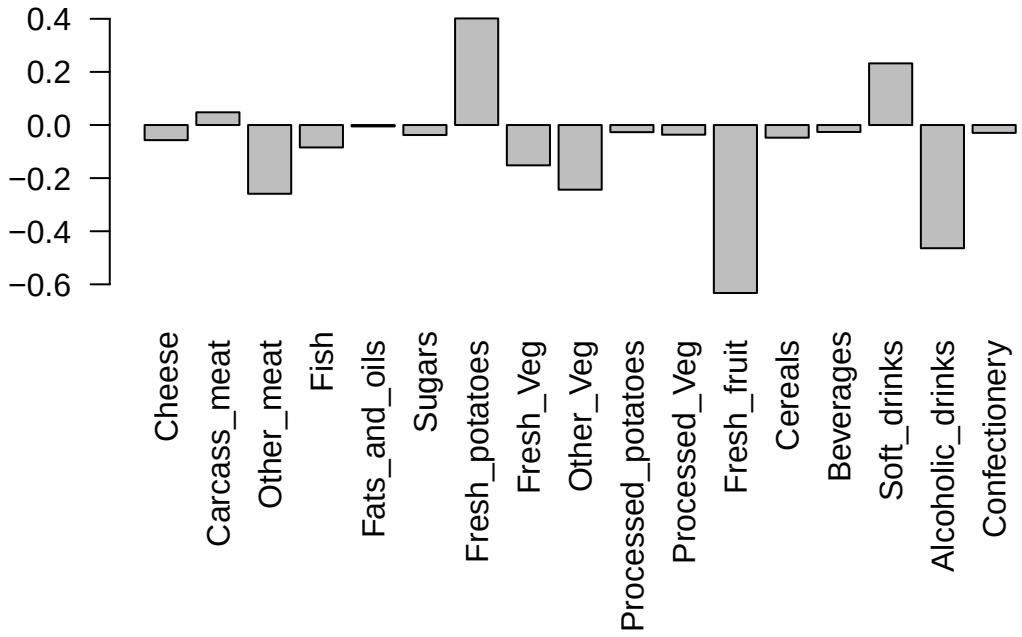
```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -4.894696e-14
Wales     -240.52915 -224.646925 -56.475555  5.700024e-13
Scotland   -91.86934  286.081786 -44.415495 -7.460785e-13
N.Ireland  477.39164  -58.901862  -4.877895  2.321303e-13
```

```
plot(pca$x[,1],pca$x[,2],
     col =c("black","red", "blue", "darkgreen"),pch=16,)
```



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 )
```


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