Lab 7: Machine Learning

Marriane Allahwerdi (A16902759)

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

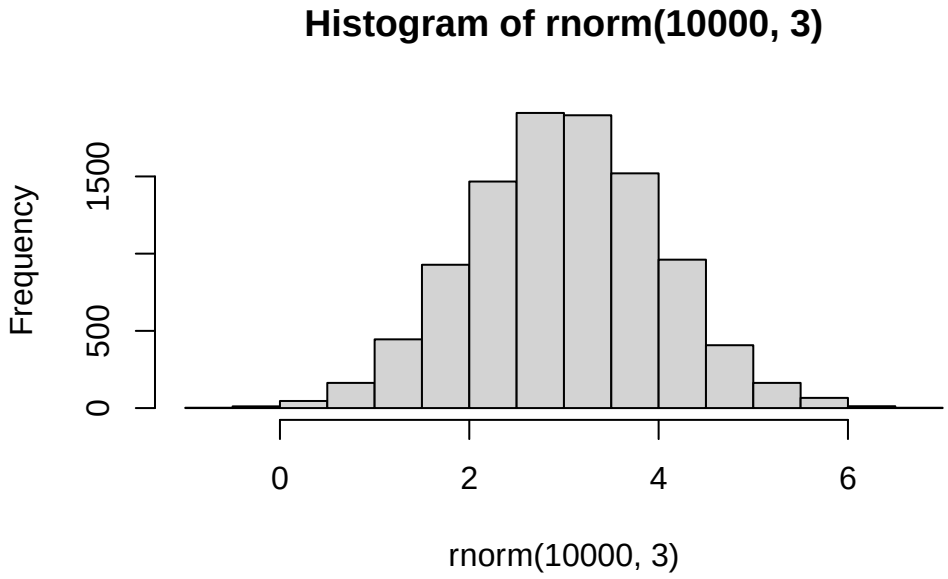
The goal here is to find groups/clusters in your input data

First I will make up some data with clear groups. For this I will use the `rnorm()` function:

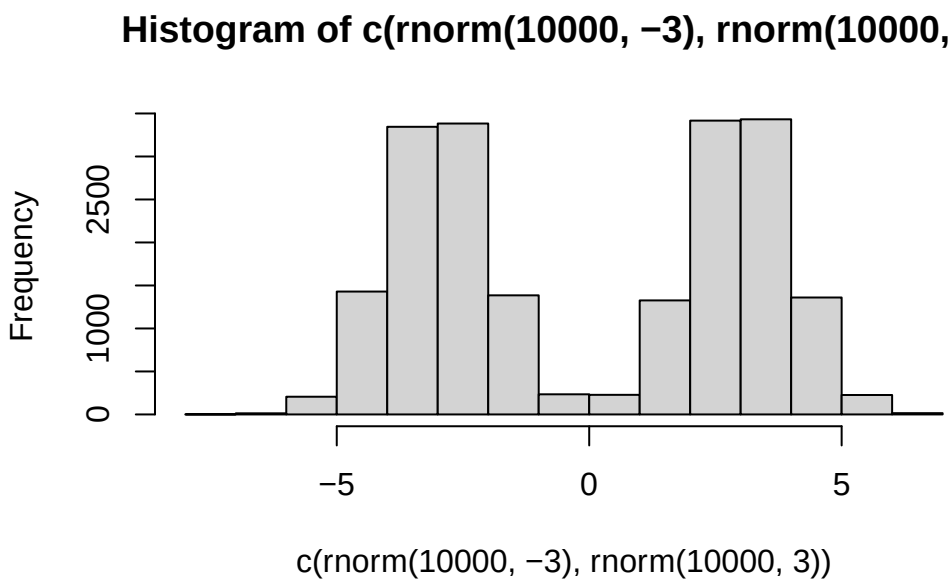
```
rnorm(10)
```

```
 [1] -0.93090697  0.69624082 -0.09737692 -0.75396804  1.04302768  1.03375008
 [7]  0.28386591  0.48993530 -0.92748377  0.24984685
```

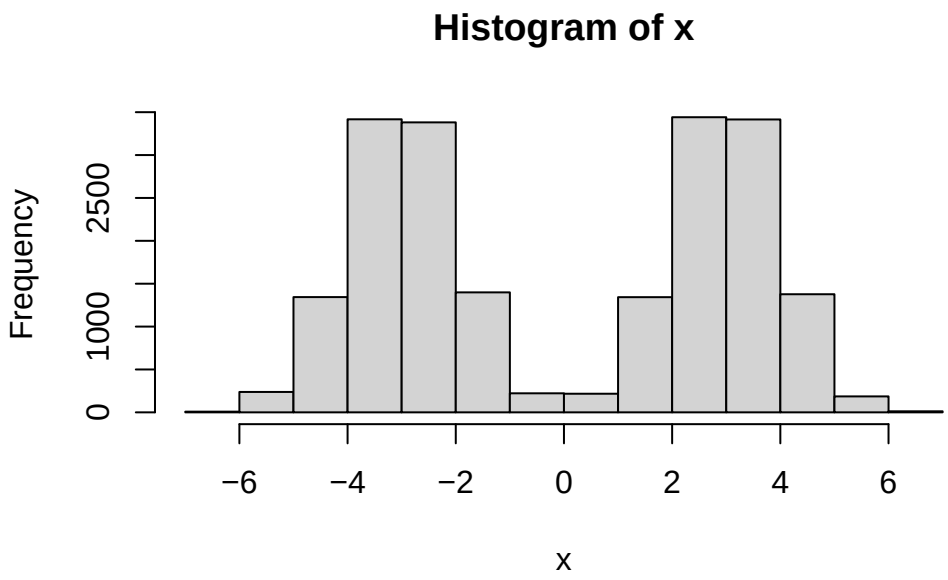
```
hist(rnorm(10000,3))
```



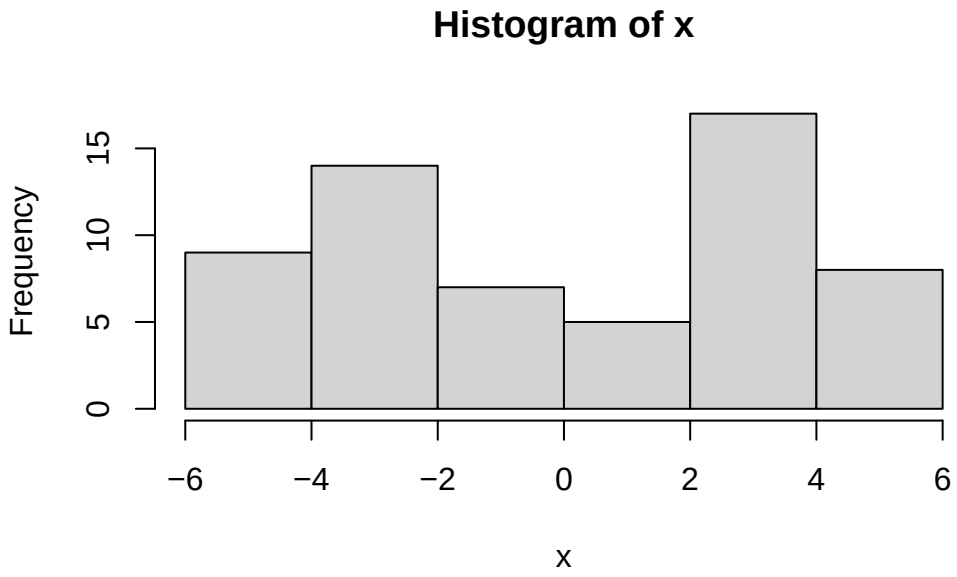
```r
hist(c(rnorm(10000, -3), rnorm(10000,3)))
```




```r
n <- 10000
x <- c(rnorm(n,-3), rnorm(n,3))
hist(x)
```



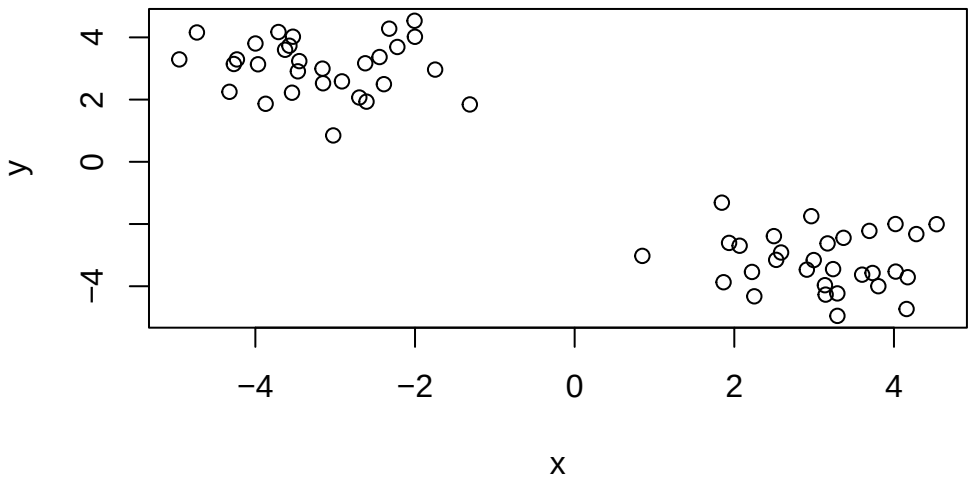

```
n <- 30
x <- c(rnorm(n,-3), rnorm(n,3))
hist(x)
```



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

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

```
             x         y
[1,] -4.231222 3.2907880
[2,] -3.024438 0.8477661
[3,] -3.152079 2.5245616
[4,] -1.313289 1.8438744
[5,] -2.221271 3.6918432
[6,] -3.159140 2.9950650
```

```
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?

30 points in each cluster, 60 total

> 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

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

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

Cluster means:
          x         y
1 -3.195796  3.071789
2  3.071789 -3.195796

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] 46.22768 46.22768
 (between_SS / total_SS =  92.7 %)

Available components:

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

Results in kmeans objects `km`

```
attributes(km)
```

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

$class
[1] "kmeans"
```

Cluster size?

```
km$size
```

```
[1] 30 30
```

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 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
```

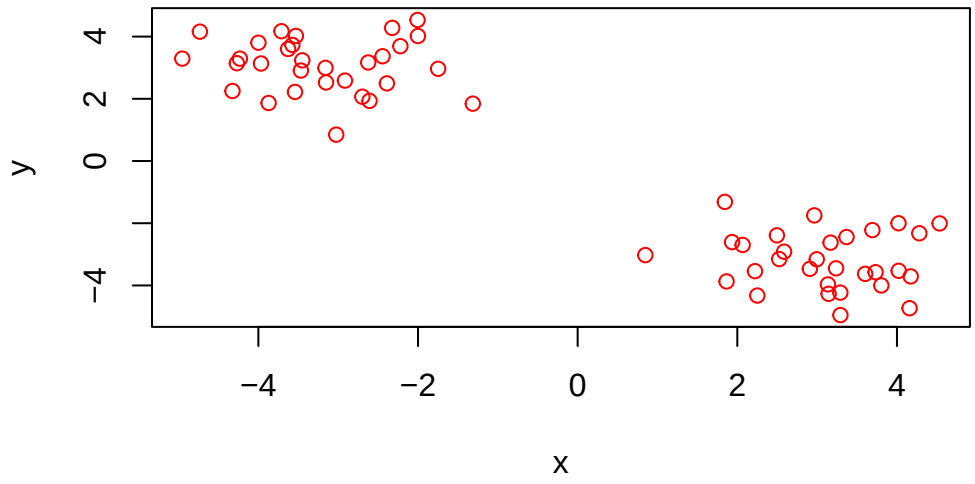
Cluster center?

```
km$center
```

```
          x         y
1 -3.195796  3.071789
2  3.071789 -3.195796
```

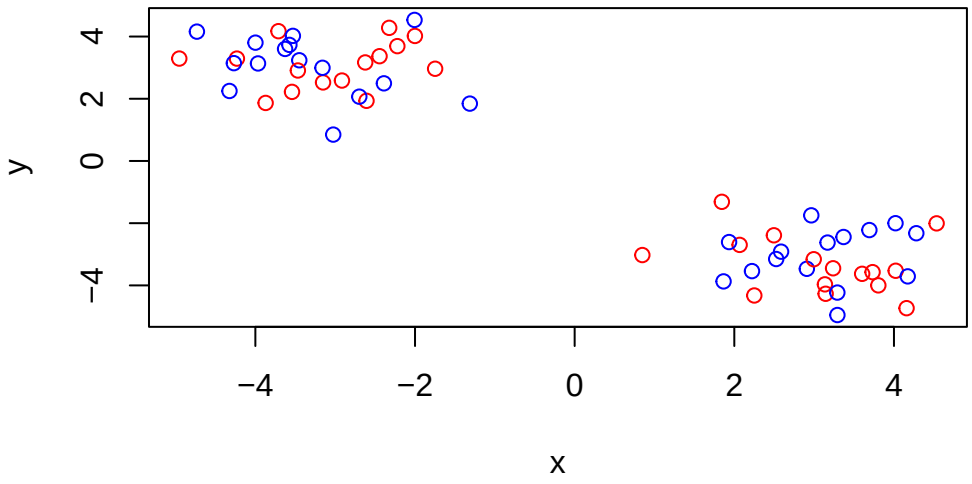
Plot x colored by the kmeans cluster assignment and add cluster centers as blue points

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

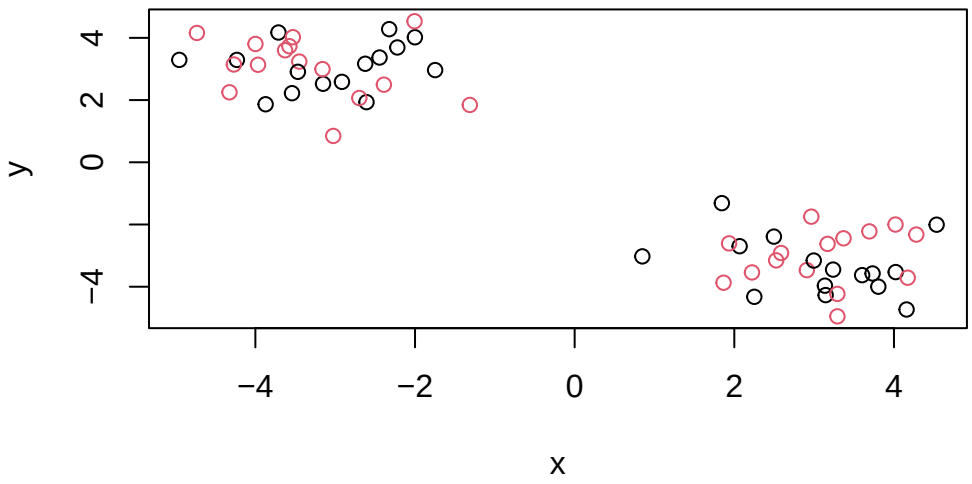


R will recycle the shorter color vector to be the same length as the longer(number of data points) in z

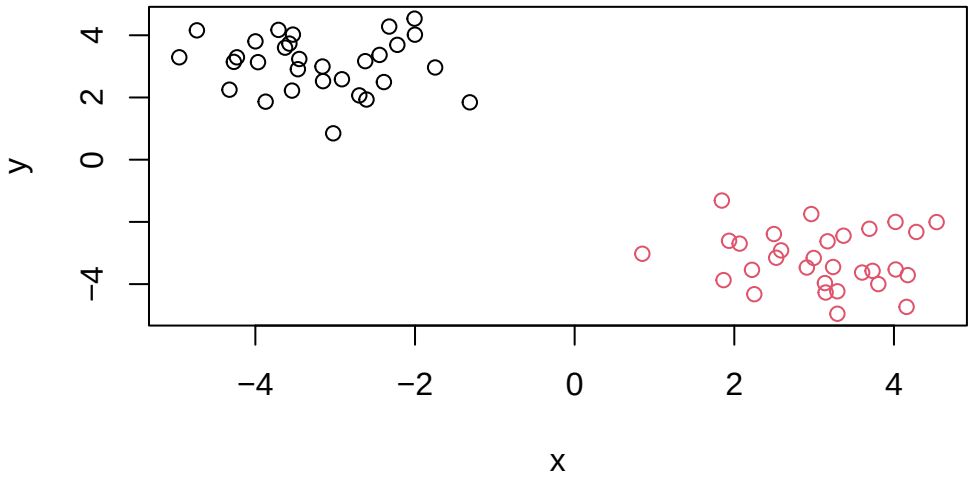
```
plot(z, col=c("red", "blue") )
```




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

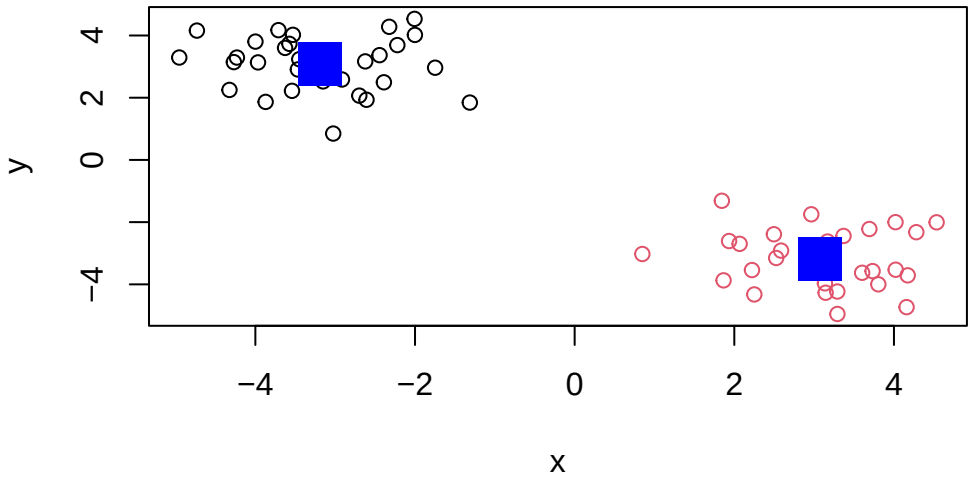



```
plot(z, col=km$cluster)
```



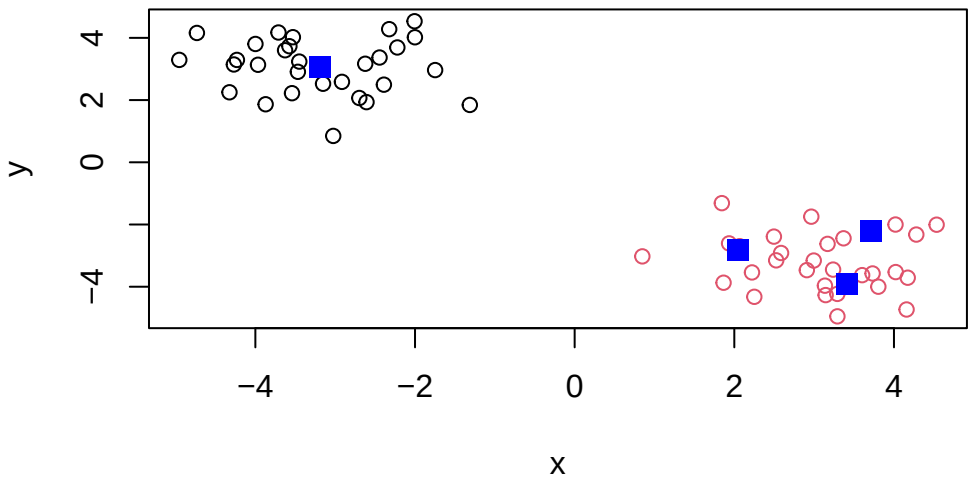
We can use the `points()` fucntion to add new points to an existing plot…like the cluster centers.

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



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=km$cluster)
points(km4$centers, col="blue", pch=15, cex=1.5)
```



Hierarchical Clustering

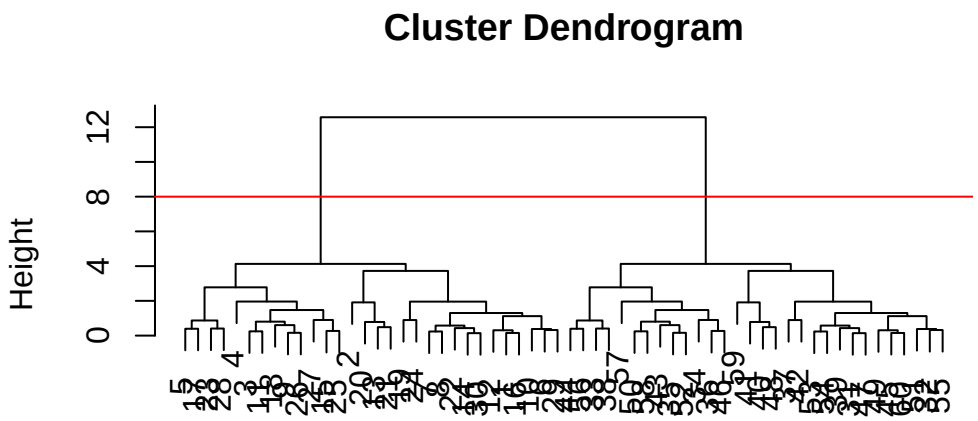
Let's take our same data **z** and see how hclust works.

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

```
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")
```



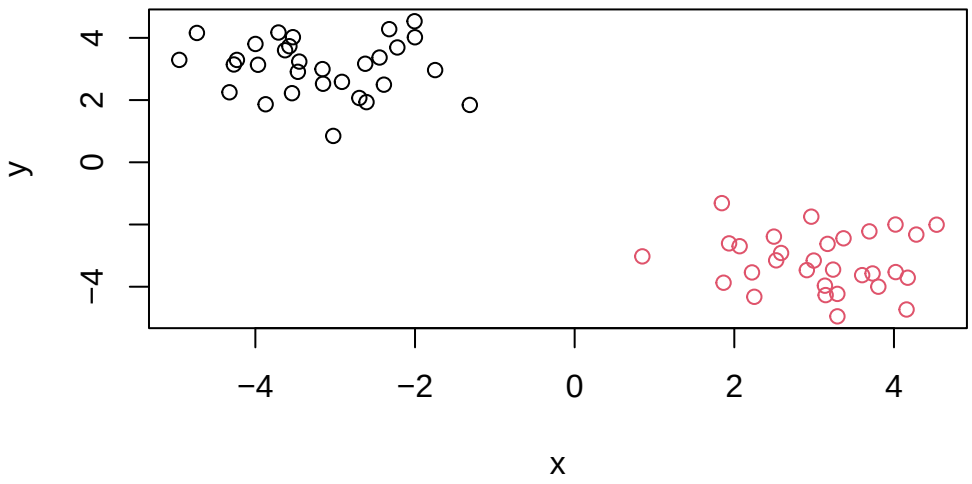
I can get my cluster membership vector by "cutting the tree" with the `cutree()` function like so:

```
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 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. Can you plot `z` colored by our hclus results:

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



LAB QUESTIONS

PCA of UK food data

```
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
```

```
dim(x)
```

```
[1] 17  4
```

Q1. How many rows and columns are in your new data frame named x? What R functions could you use to answer this questions?

There are 17 rows and 4 columns.

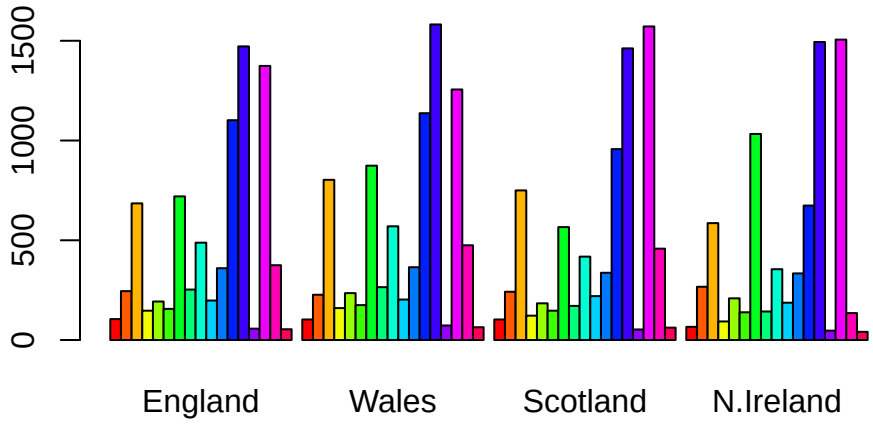
```
x <- read.csv(url, row.names=1)
head(x)
```

```
              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 method because it is more generalized than the first dataset. The other code shows different parts of the data when run again while the first method is more clear in showing the information.

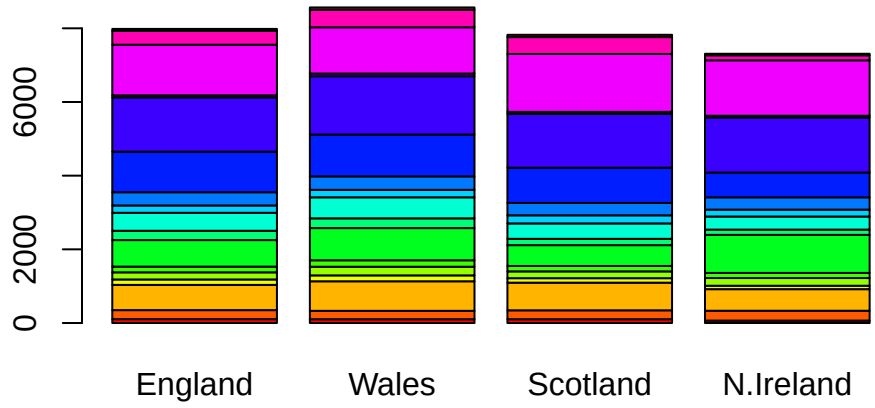
```
barplot(as.matrix(x), beside=T, col=rainbow(nrow(x)))
```



Q3: Changing what optional argument in the above barplot() function results in the following plot?

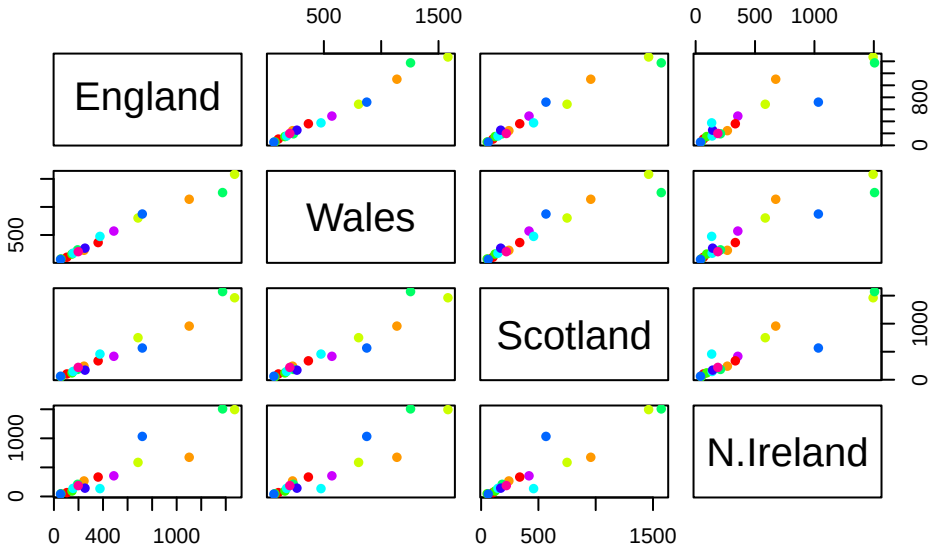
Changing Beside=T to Beside=F as seen below.

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



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

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



> 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?

A `pairs()` function creates a scatterplot matrix that can be used to visualize pairwise relationships in the dataset. The diagonal represents each variable plotted against itself, these points are not that valuable for analysis.

> Q6. What is the main differences between N. Ireland and the other countries of the UK in terms of this data-set?

It is hard to see structure and trends in even this small data-set. We can't tell what it means if a point lies on the diagonal compared for a given plot.

PCA to the rescue

Lets see how PCA deals with this dataset. 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 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 this `pca` object that we created from running `prcomp`

```
attributes(pca)
```

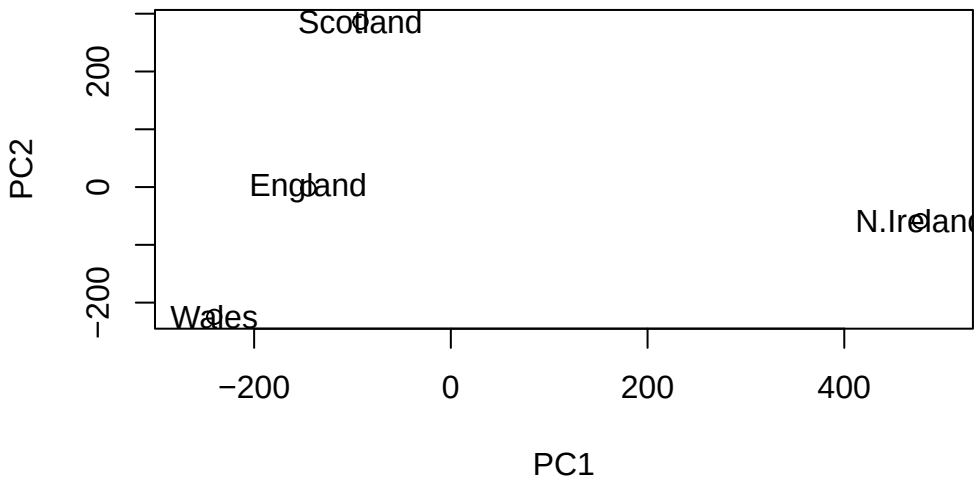
```
$names
[1] "sdev"     "rotation" "center"   "scale"    "x"

$class
[1] "prcomp"
```

```
pca$x
```

```
                PC1         PC2        PC3           PC4
England   -144.99315   -2.532999 105.768945 -9.152022e-15
Wales     -240.52915 -224.646925 -56.475555  5.560040e-13
Scotland   -91.86934  286.081786 -44.415495 -6.638419e-13
N.Ireland  477.39164  -58.901862  -4.877895  1.329771e-13
```

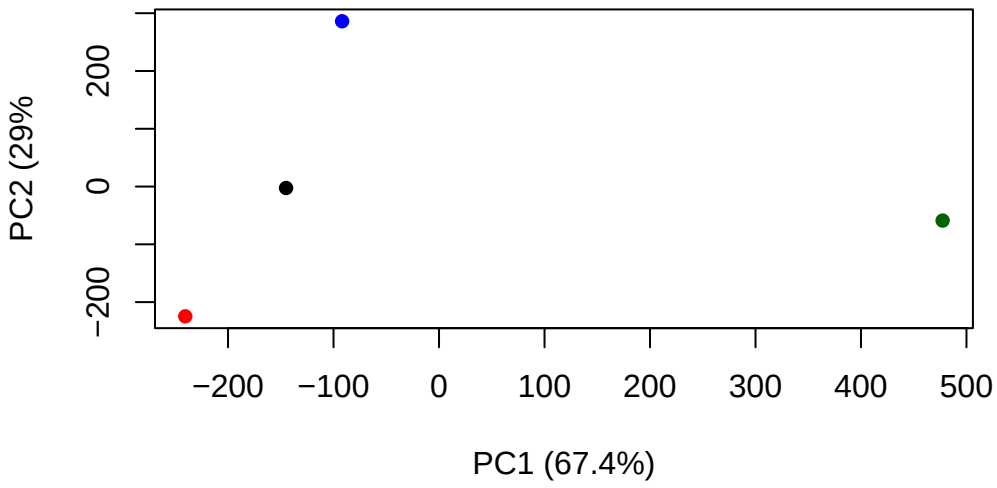
Q7. Complete the code below to generate a plot of PC1 vs PC2. The second line adds text labels over the data points.

```
# 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(pca$x[,1], pca$x[,2], col=c("black", "red", "blue", "darkgreen"), pch=16, xlab="PC1 (67
```



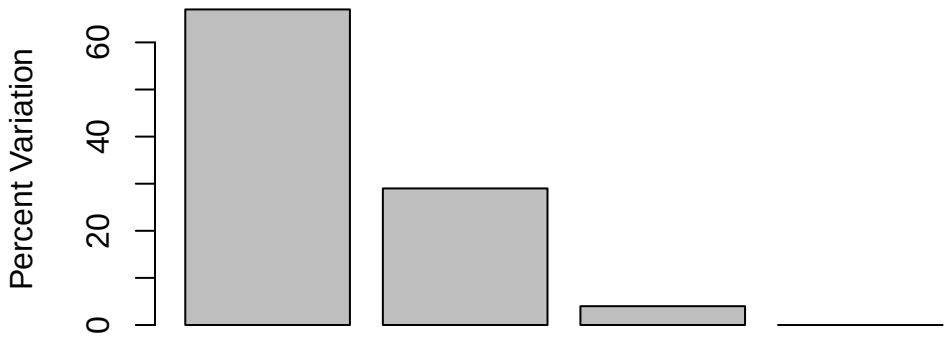

```
v <- round( pca$sdev^2/sum(pca$sdev^2) * 100 )
v
```

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

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

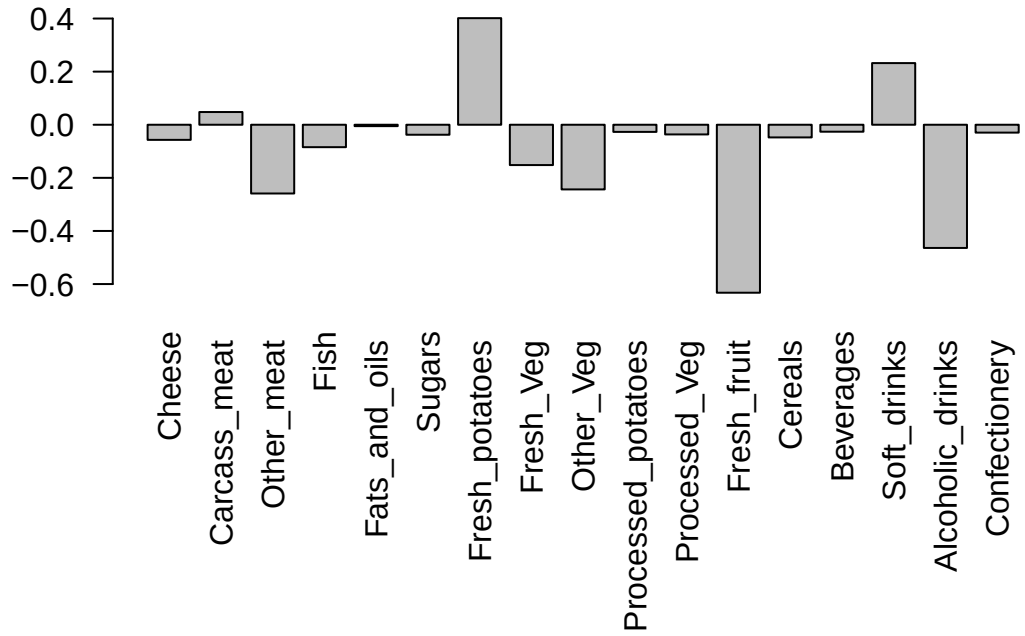
```
                             PC1       PC2      PC3           PC4
Standard deviation     324.15019 212.74780 73.87622 2.921348e-14
Proportion of Variance   0.67444   0.29052  0.03503 0.000000e+00
Cumulative Proportion    0.67444   0.96497  1.00000 1.000000e+00
```

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



Principal Component

```
## 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 )
```



Q9: Generate a similar 'loadings plot' for PC2. What two food groups feature prominantely and what does PC2 mainly tell us about?

The 2 main food groups featured are fresh_potatoes among and soft_drinks. PC2 mainly tells us the second most degree of variation among the countries which appears between those two food groups as seen in the graph below.

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