Class 7: Machine Learning 1

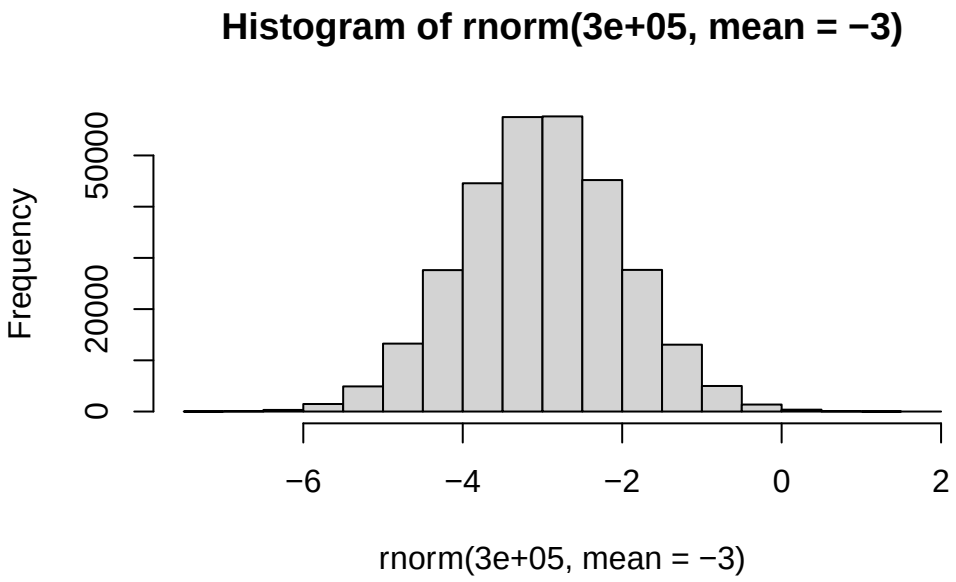
Challana Tea

In this class, we will explore and get practice with clustering and Principle Component Analysis (PCA)

#Clustering with K-means

First we will make up some data to cluster where we know what the result should be.

```
hist(rnorm(300000, mean=-3))
```



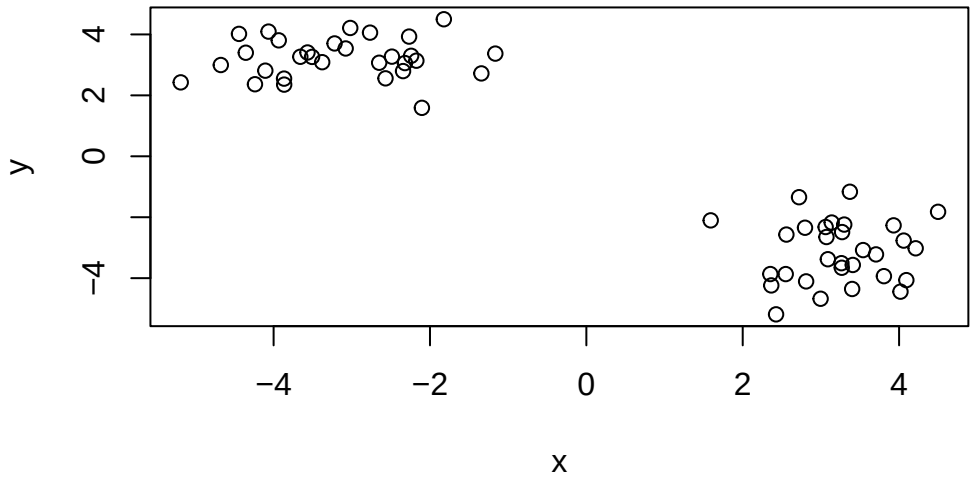
I want a little vector with two groupings in it:

```r
tmp <- c(rnorm(30, -3), rnorm(30, +3))
x <- data.frame(x=tmp, y=rev(tmp))
head(x)
```

```
          x        y
1 -2.320753 3.060642
2 -2.766857 4.059233
3 -3.658296 3.267114
4 -3.866532 2.550075
5 -3.566665 3.407803
6 -3.019821 4.212135
```

Let's have a look:

```r
plot(x)
```




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

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

```
Cluster means:
         x        y
1 -3.14739  3.22222
2  3.22222 -3.14739

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 2

Within cluster sum of squares by cluster:
[1] 42.33726 42.33726
 (between_SS / total_SS =  93.5 %)

Available components:

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

It is important to not just run the analysis but to be able to get your important results back in a way that we can do things with them.

Q. How do I find the cluster sizes

```
km$size
```

```
[1] 30 30
```

Q. How about the cluster centers?

```
km$centers
```

```
         x        y
1 -3.14739  3.22222
2  3.22222 -3.14739
```

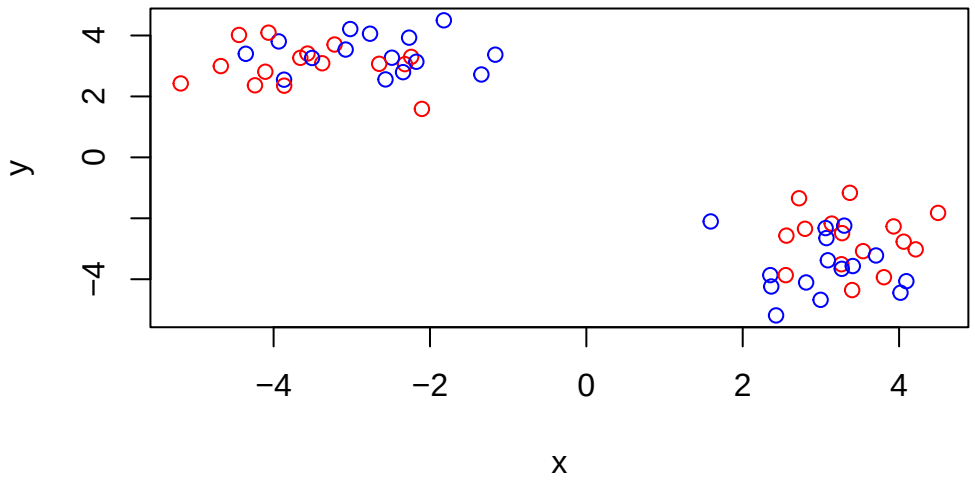
Q. How about the main result - the cluster assignment vector?

```
km$cluster
```

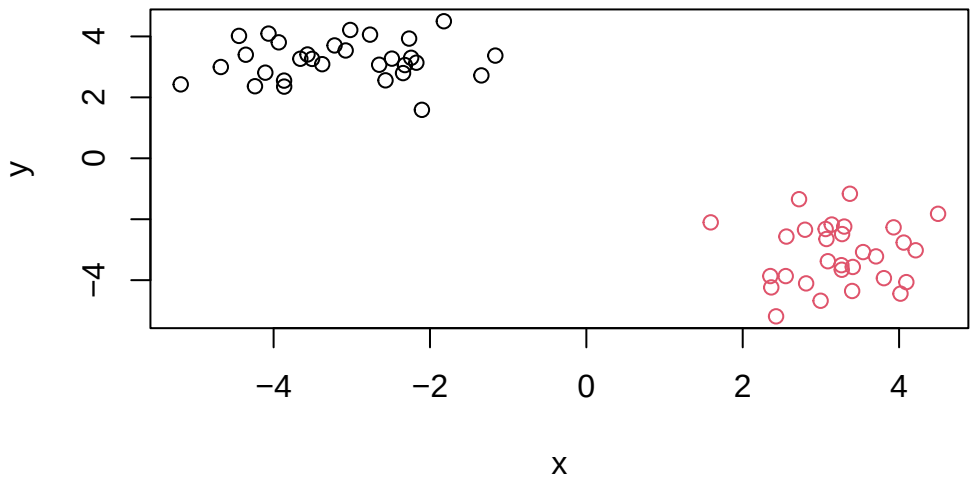
```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2
[39] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
```

Q. Can we make a summary figure showing our clustering result? - The points colored by cluster assignment and maybe add the cluster centers as a different color?

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

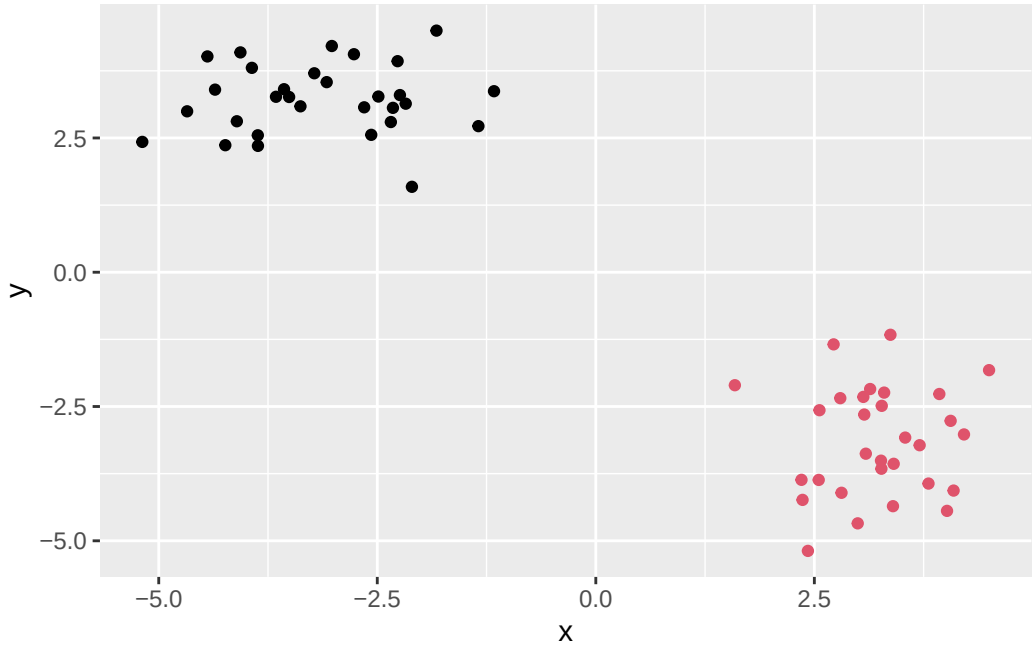



```
plot(x, col=km$cluster)
```



I need 3 things: data, aes, geoms

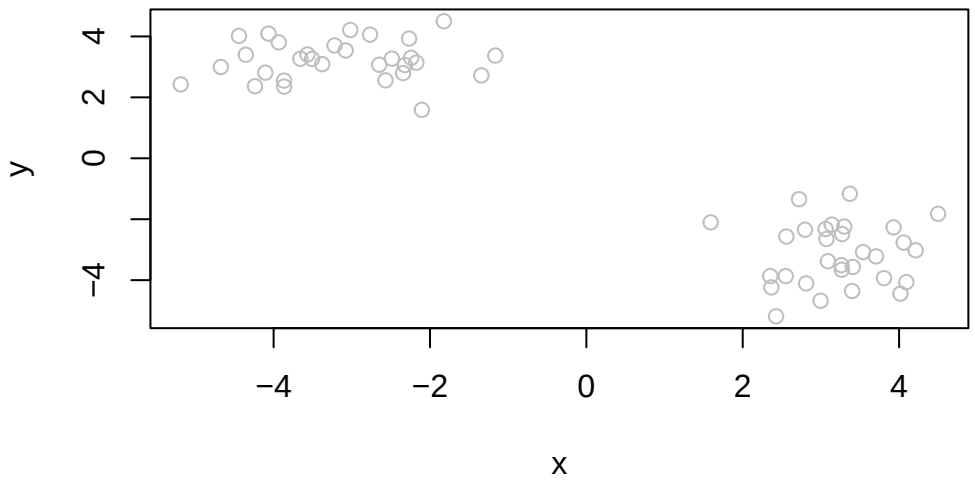
```
library(ggplot2)
ggplot(x) +
  aes(x,y) +
  geom_point(col=km$cl)
```




```
# Make up a color vector
mycols <- rep("gray", 60)
mycols
```

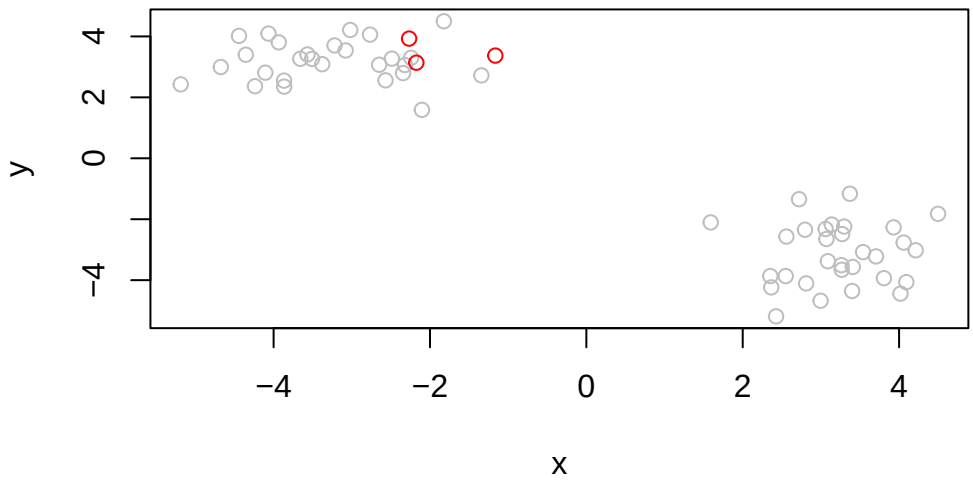
```
 [1] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[11] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[21] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[31] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[41] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
[51] "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray" "gray"
```

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



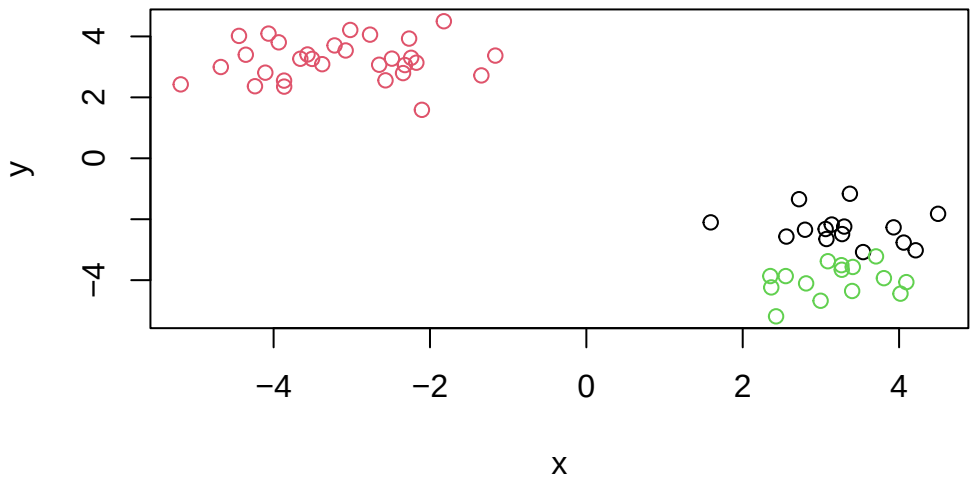
Let's highlight points 10, 12 and 20

```
mycols[c(10,12,20)] <- "red"
plot(x, col=mycols)
```



Play with different numbers and centers

```
km <- kmeans(x, centers=3)
plot(x, col=km$cluster)
```




```
km$tot.withinss
```

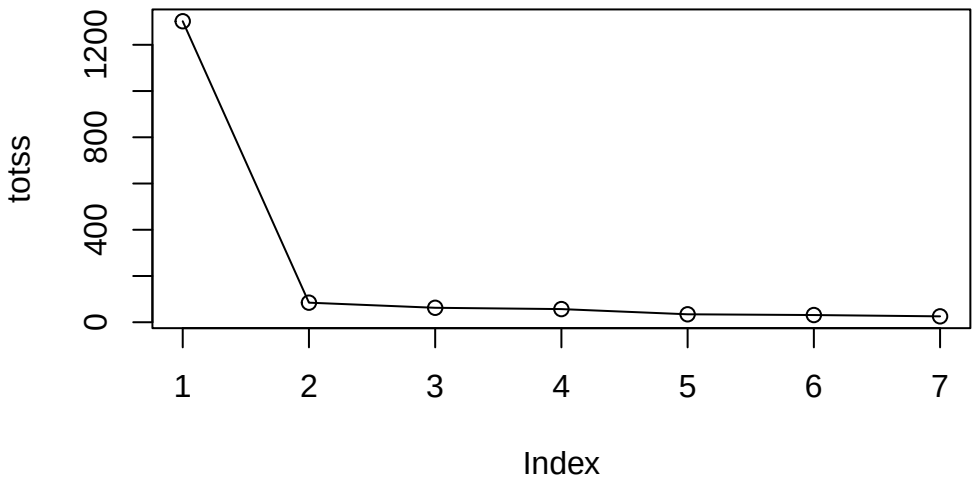
```
[1] 62.54373
```

What we want to do is try out different numbers of K from 1 to 7. We can write a `for` loop to do this for us and store the `$tot.withinss` each time.

```
totss <- NULL
k <- 1:7

for(i in k) {
  totss <- c(totss,kmeans(x, centers=i)$tot.withinss)
}
```

```
plot(totss, typ="o")
```



#Hierarchical Clustering

We can not just give the `hclust()` function of input data `x` like we did for the `kmeans()`.

We need to first calculate a "distance matrix". The `dist()` function by default will calculate euclidean distance.

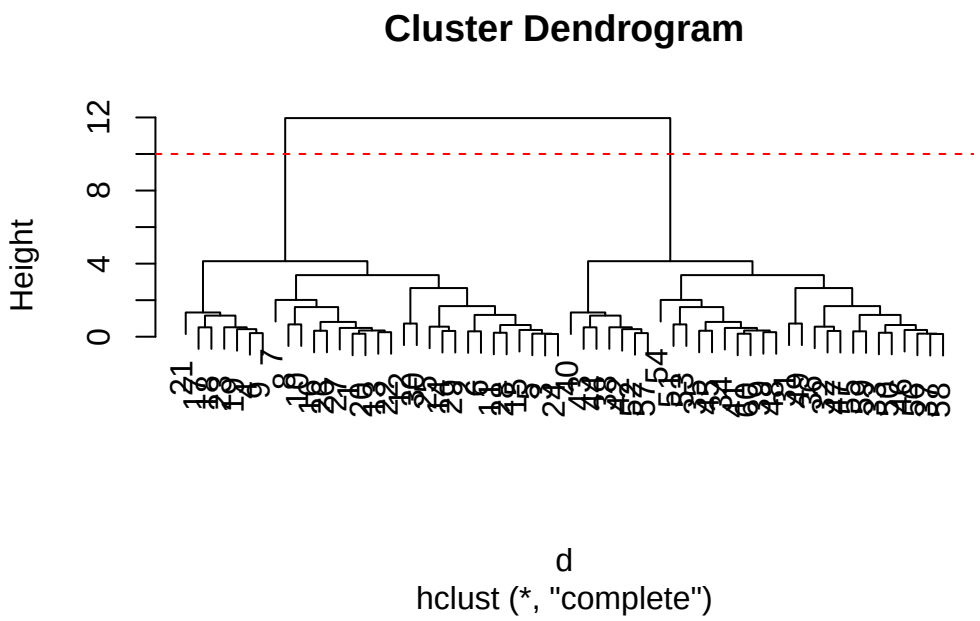
```
d <- dist(x)
hc <- hclust(d)
hc
```

```
Call:
hclust(d = d)

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

The print out is not very helpful, but the plot method is useful

```
plot(hc)
abline(h=10, col="red", lty=2)
```



To get my all important cluster membership vector out of a hclust object I can use the `cutree()`

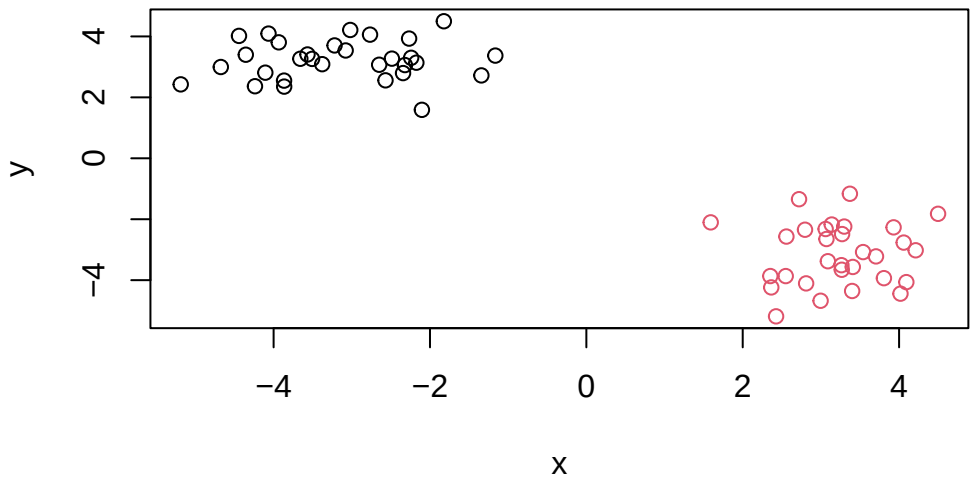
```
cutree(hc,h=10)
```

```
 [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
```

You can also set a `k=` argument to `cutree()`

```
grps <- cutree(hc, k=2)
```

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


y
4
2
0
−4
−4
−2
0
2
4
x