Class 7: Machine Learning 1

Dylan Mullaney

Today we will explore unsupervised machine learning methods uncluding clustering and dimensionality reduction methods.

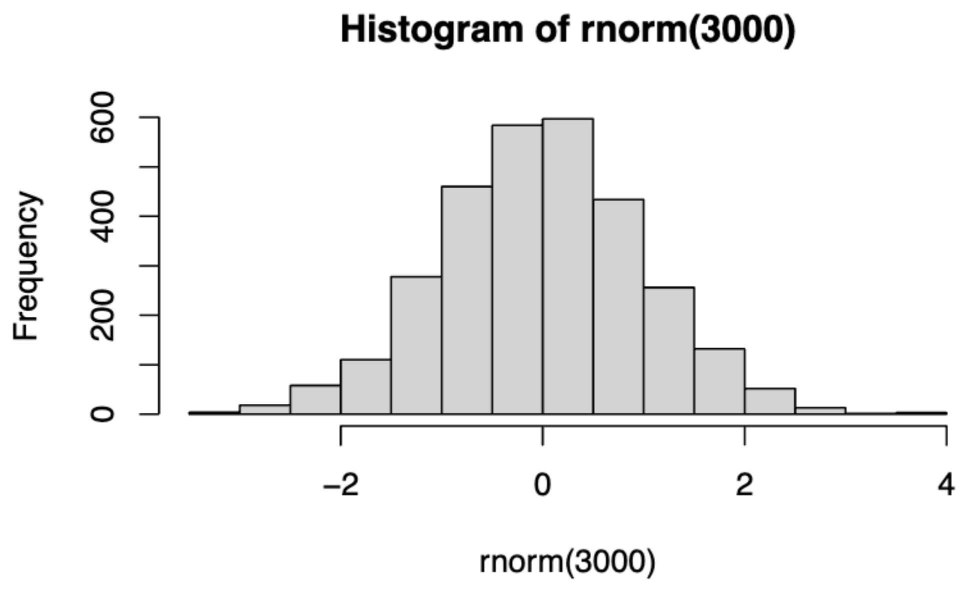
Let's start by making up some data (where we know there are clear groups) that we can use to test out different clustering methods.

We can use the 'rnorm()' function to help us here: - has three input values (n, mean, sd) - mean and sd have defaults, set as equals to a value

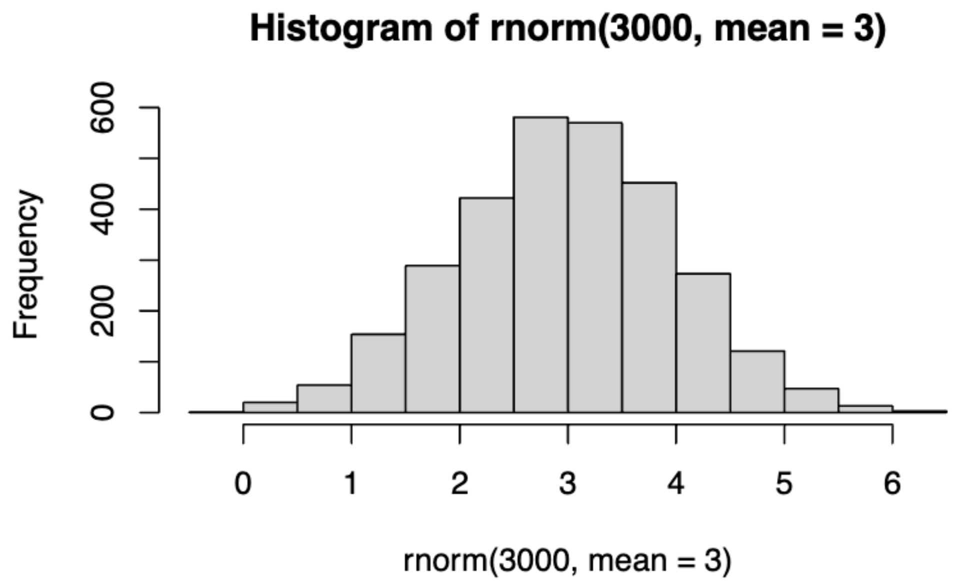
```
#rnorm(n, mean=0, sd=1)
rnorm (30)
```

```
 [1] -1.03192247  1.21292171 -0.74529963  0.22083389 -1.18942372 -1.47944141
 [7]  1.34563276 -0.59998216  1.44376858  0.05702675  0.44802885 -0.29251310
[13]  0.85602333 -1.28179762 -1.61456234  0.98184717 -1.01885980 -0.14850902
[19]  0.50489980 -1.94343592 -1.69048756  0.13124353 -0.54197562  0.16314493
[25] -0.15983905  0.78518439  1.65036954 -0.85103283  0.65734648  0.59960571
```

```
hist(rnorm(3000))
```



```
hist(rnorm(3000, mean=3))
```

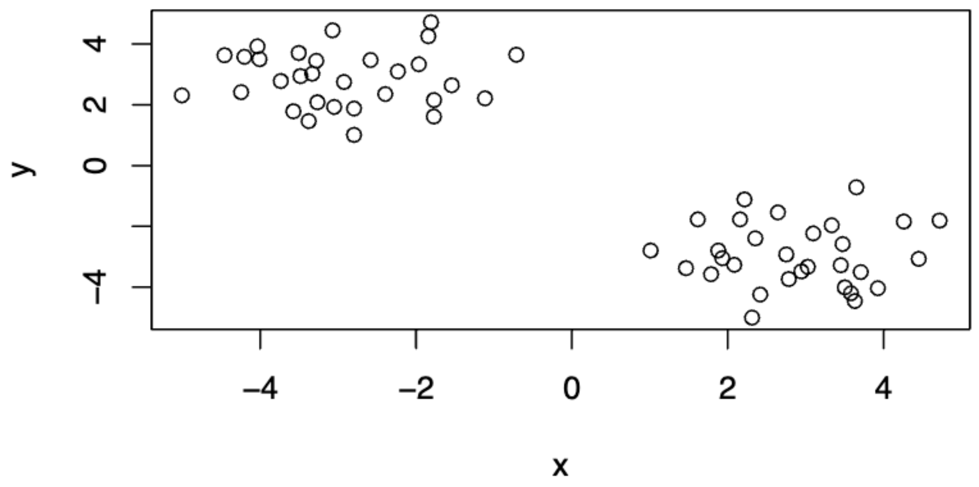


Make data with two "clusters". Use c and a comma to combine these two vector sets of data.

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

z<- cbind(x=x, y=rev(x)) ##rev reverses the data
head(z)
```

```
            x          y
[1,] 1.463818 -3.3781789
[2,] 4.260020 -1.8425167
[3,] 3.097958 -2.2330163
[4,] 3.502588 -4.0093902
[5,] 3.650511 -0.7132084
[6,] 2.214773 -1.1145722
```

```r
plot(z)
```



##K-means clustering

The main function in "base" R for K-means clustering is called 'kmeans()'

There are 6 inputs: - Only two don't have defaults, 'x' and 'centers'

```r
k<- kmeans(z, 2)
k
```

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

Cluster means:
          x         y
1 -2.929724  2.872160
2  2.872160 -2.929724

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 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] 57.07592 57.07592
 (between_SS / total_SS =  89.8 %)

Available components:

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

What are the details of the data?

```r
attributes(k)
```

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

$class
[1] "kmeans"
```

Q. How many points lie in each cluster?

```r
k$size
```

```
[1] 30 30
```

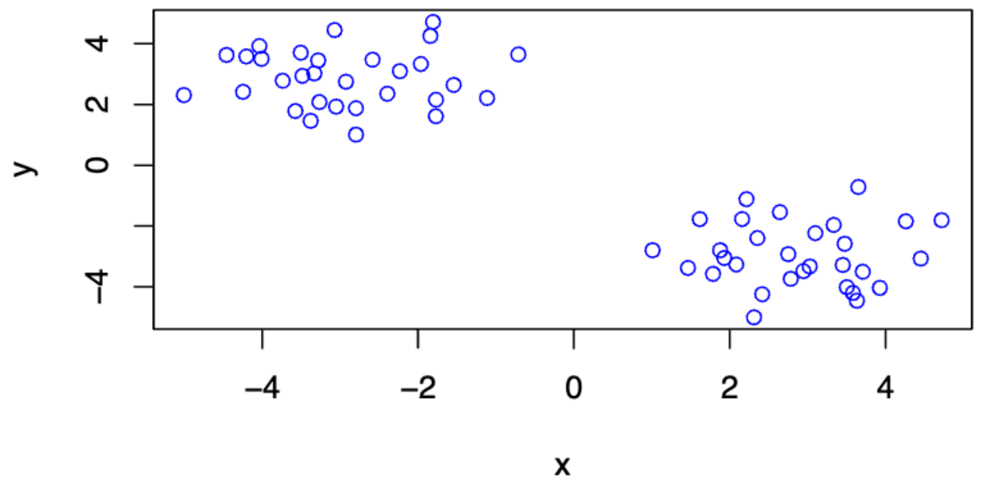
Q. What component of ous results tells us about cluster membership? –> which point is in which cluster

```r
k$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 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
```

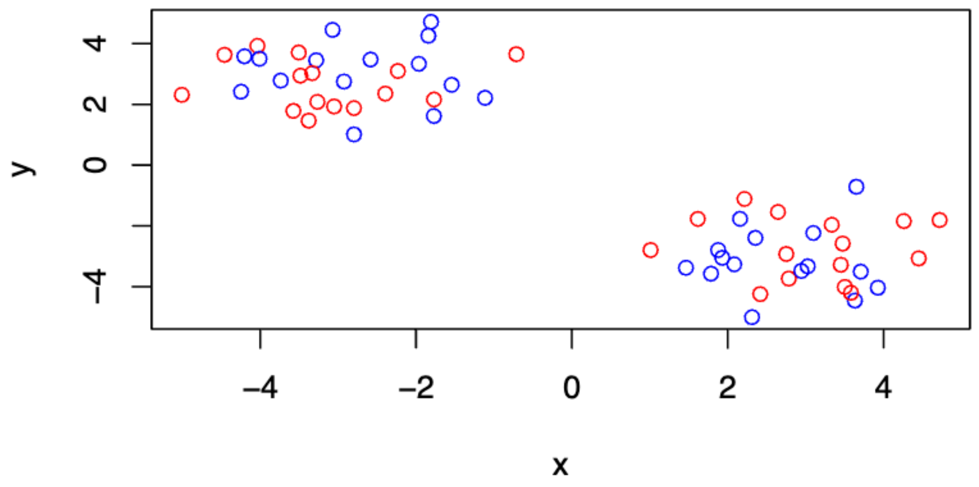
Q. Put this result info together and make a "base R" plot of our clustering results. Add the cluster center points to this plot.

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



I want each of the clusters to be colored differently.

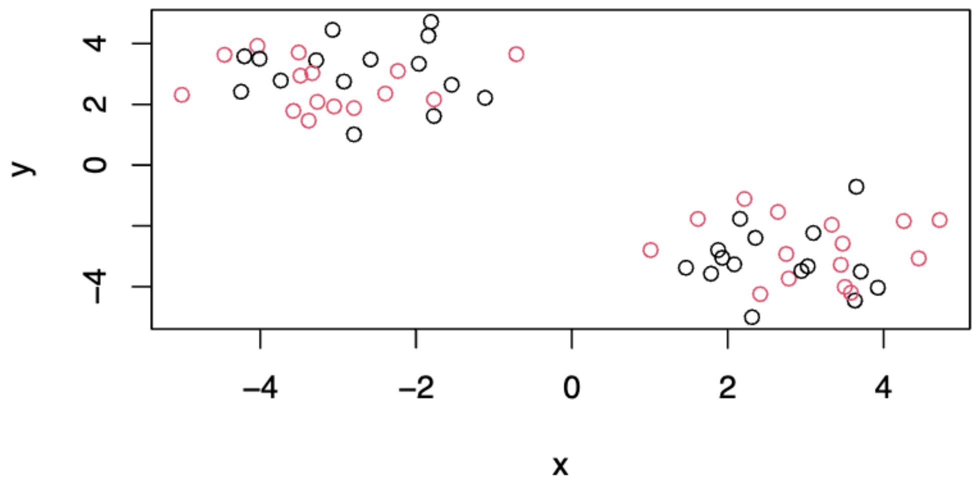
```r
plot(z, col=c("blue", "red"))
```



```
# Will alternate back and forth between both colors
```

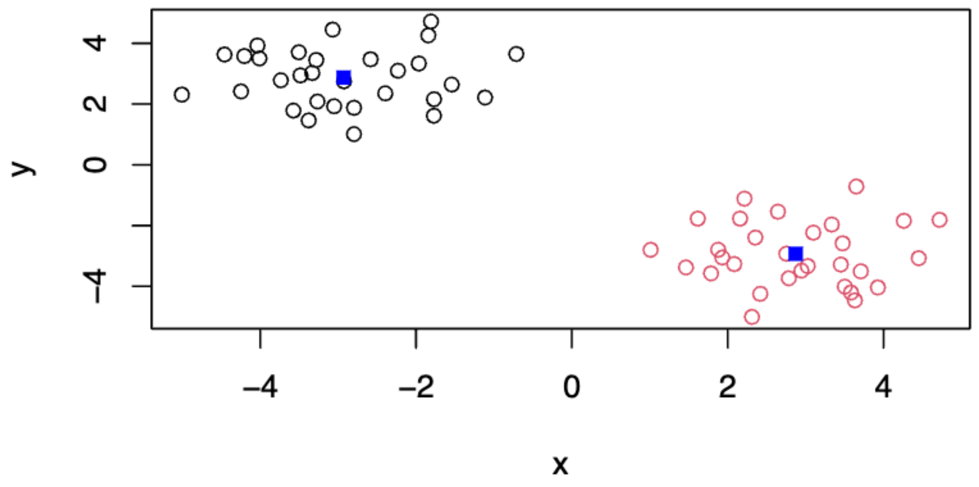
You can color by number in the data set.

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



Plot colored by cluster membership -pch is plotting character to change the type of dot

```r
plot(z, col=k$cluster)
 points(k$centers, col="blue", pch=15)
```

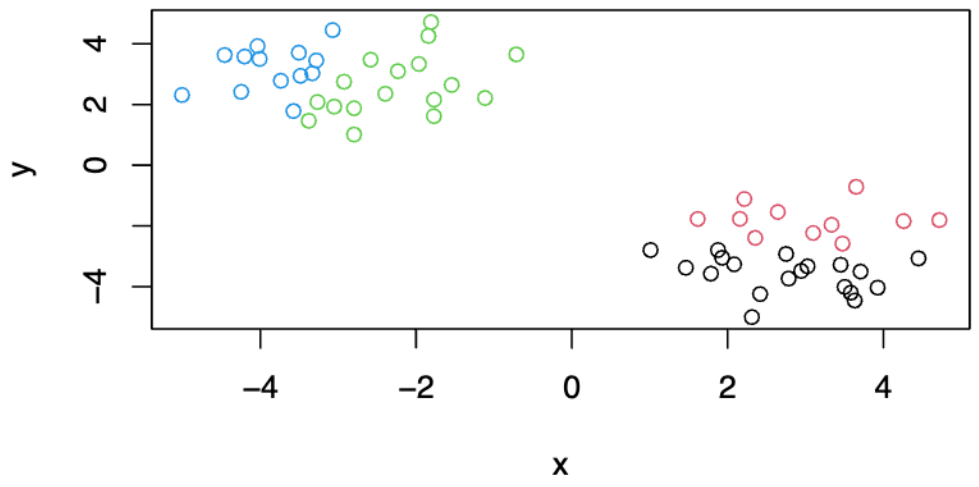


Q. Run kmeans on our input 'z' and define 4 clusters makes the same result visualization plot as above - plot colored by cluster membership.

```r
k4<- kmeans(z, 4)
k4
```

```
K-means clustering with 4 clusters of sizes 19, 11, 17, 13

Cluster means:
          x         y
1  2.770526 -3.587073
2  3.047710 -1.794303
3 -2.232356  2.625977
4 -3.841666  3.194091

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

Within cluster sum of squares by cluster:
[1] 22.17427 11.97536 25.52475 10.09478
 (between_SS / total_SS =  93.8 %)

Available components:

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

```r
plot(z, col=k4$cluster)
```



Is this good clustering? No not really but how do we know

```
k4$withinss
```

```
[1] 22.17427 11.97536 25.52475 10.09478
```

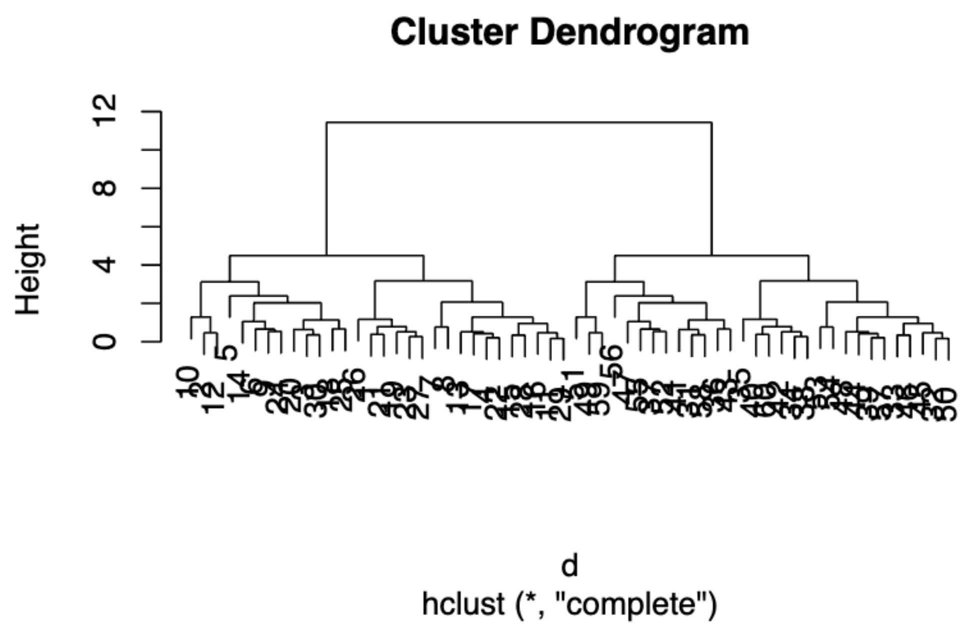
Hierarchical Clustering –> hclust()

The main function in base R is called hclust(). It will take as input a distance matrix (key point is that you can't just give your "raw" data as input - you have to first calculate a distance matrix from your data).

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

```
Call:
hclust(d = d)

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

```
plot(hc)
```

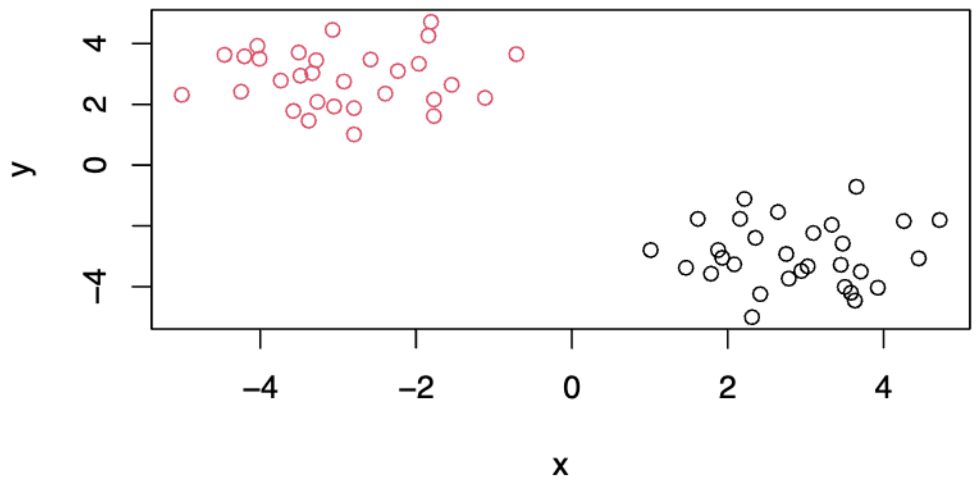


Once I inspect the "tree", I can "cut" the tree to yield my groupings/clusters. The function to do this is called 'cutree()'.

```
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 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
```

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



Dimensionality Reduction - Principal Component Analysis (PCA)

- Objective: reduce the features dimensionality whie only losing a small amount of information
- Visualize multidimensional data more reasonably
- Use as a filter yp pass to other machine learning methods
- Big funnel, lots of data in, something we can make sense of out