hier_kmeans_dbscan_cluster_experiments.R

vikaskamath

2021-03-21

```
## Attempting some clustering techniques on the NCI 160 cancer data set
## Hierarchical clustering + K Means + DBSCAN

#install.packages("ISLR")
library(ISLR)

### Read the description of the data:

?NCI60
nci_labels = NCI60$labs
nci_data   = NCI60$data

dim(nci_data)
```

```
## [1]   64 6830
```

```
class(nci_labels)
```

```
## [1] "character"
```

```
class(nci_data)
```

```
## [1] "matrix" "array"
```

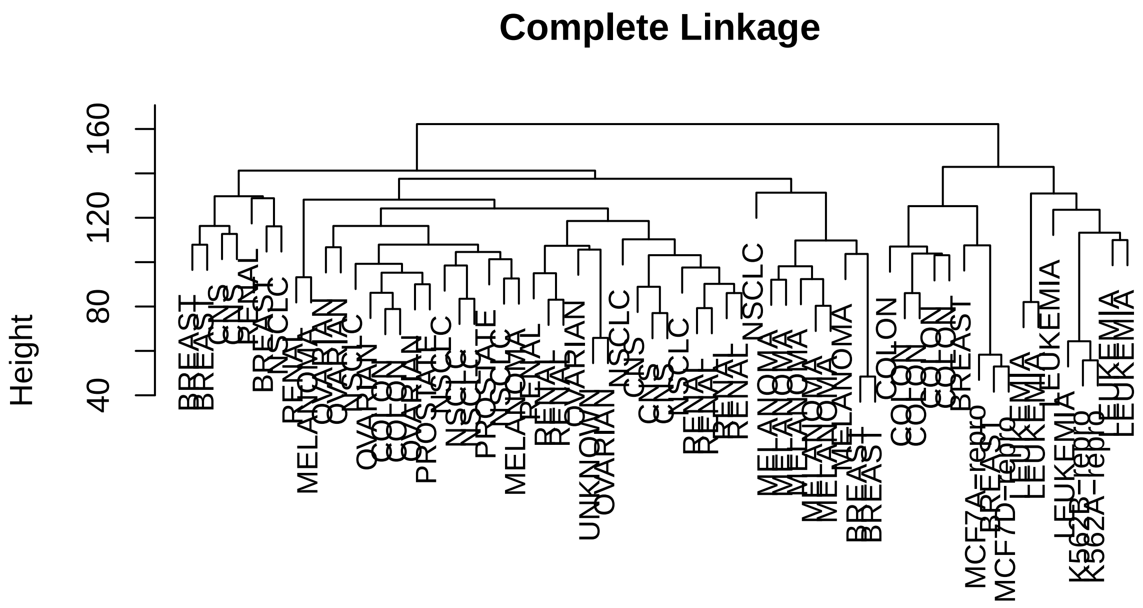
```
### The following scales the data column wise:

x = scale(nci_data)

### Find the distances:

dist_x  = dist(x)

### Write the distances to a CSV file
write.csv(as.matrix(dist_x),"distmat.csv", row.names = TRUE)

## Whats the max distance between any two pairs of vectors and which are those vectors??
which(as.matrix(dist_x) == max(as.matrix(dist_x)), arr.ind = TRUE)
```

```
##     row col
## V39  39   5
## V5    5  39
```

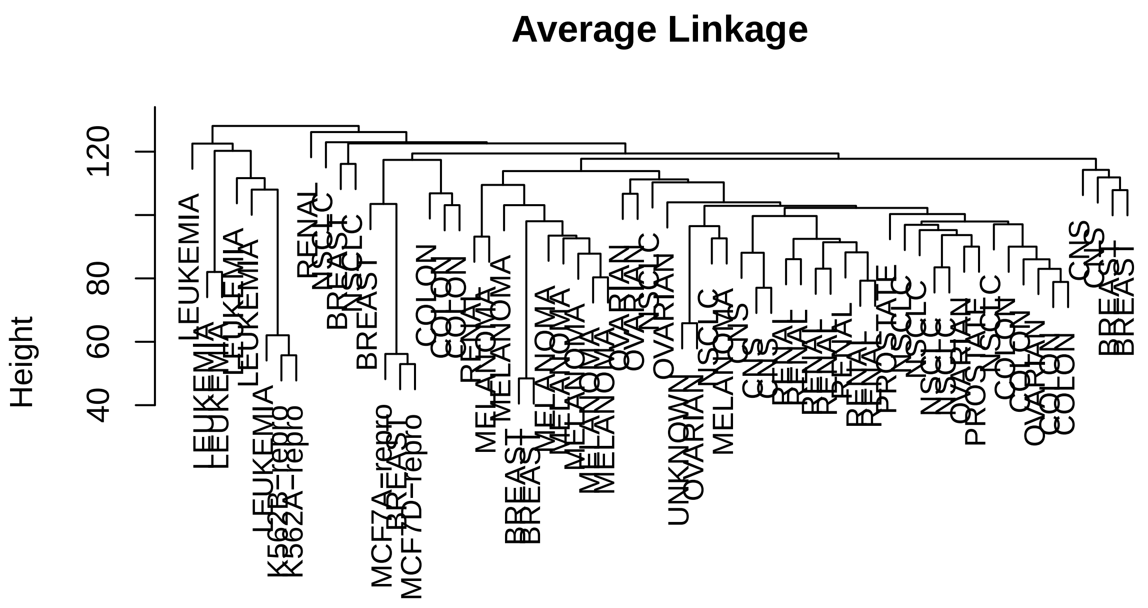
```
max(as.matrix(dist_x))
```

```
## [1] 162.2074
```

```
## Lets try all types of Hierarchical clustering
hc_complete = hclust(dist_x, method = "complete")
hc_average  = hclust(dist_x, method = "average")
hc_single   = hclust(dist_x, method = "single")

### The dendrograms look better if you display them individually:

plot(hc_complete ,main = "Complete Linkage", xlab="", sub ="", cex =.9, labels = nci_labels)
```



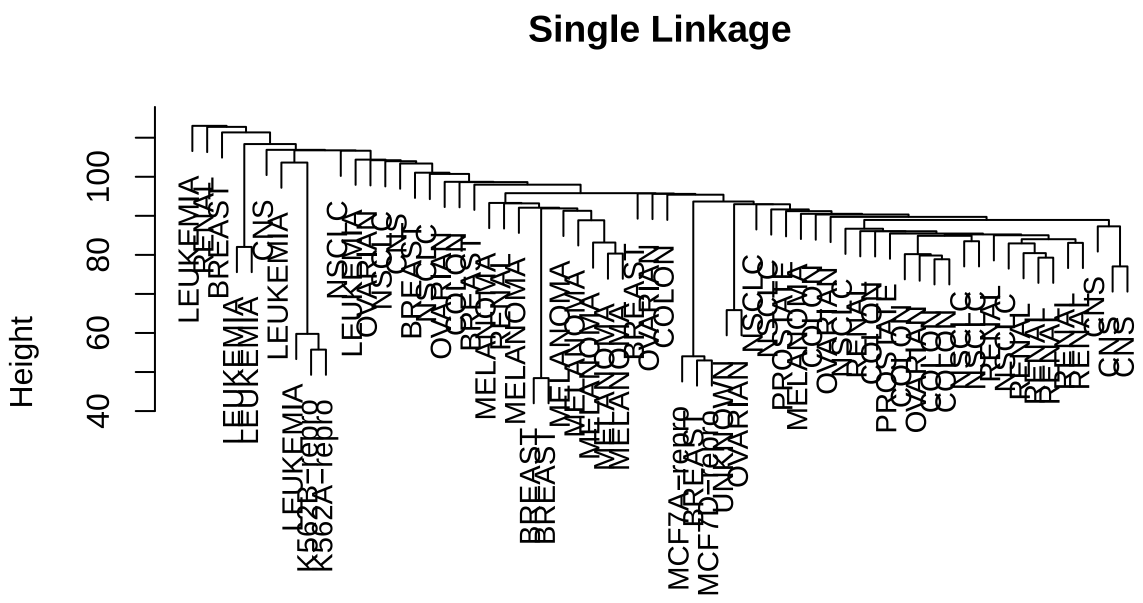
```
plot(hc_average , main = "Average Linkage",  xlab="", sub ="", cex =.9, labels = nci_labels)
```



```
plot(hc_single ,  main = "Single Linkage",   xlab="", sub ="", cex =.9, labels = nci_labels)

#install.packages("factoextra")
library(factoextra)
```

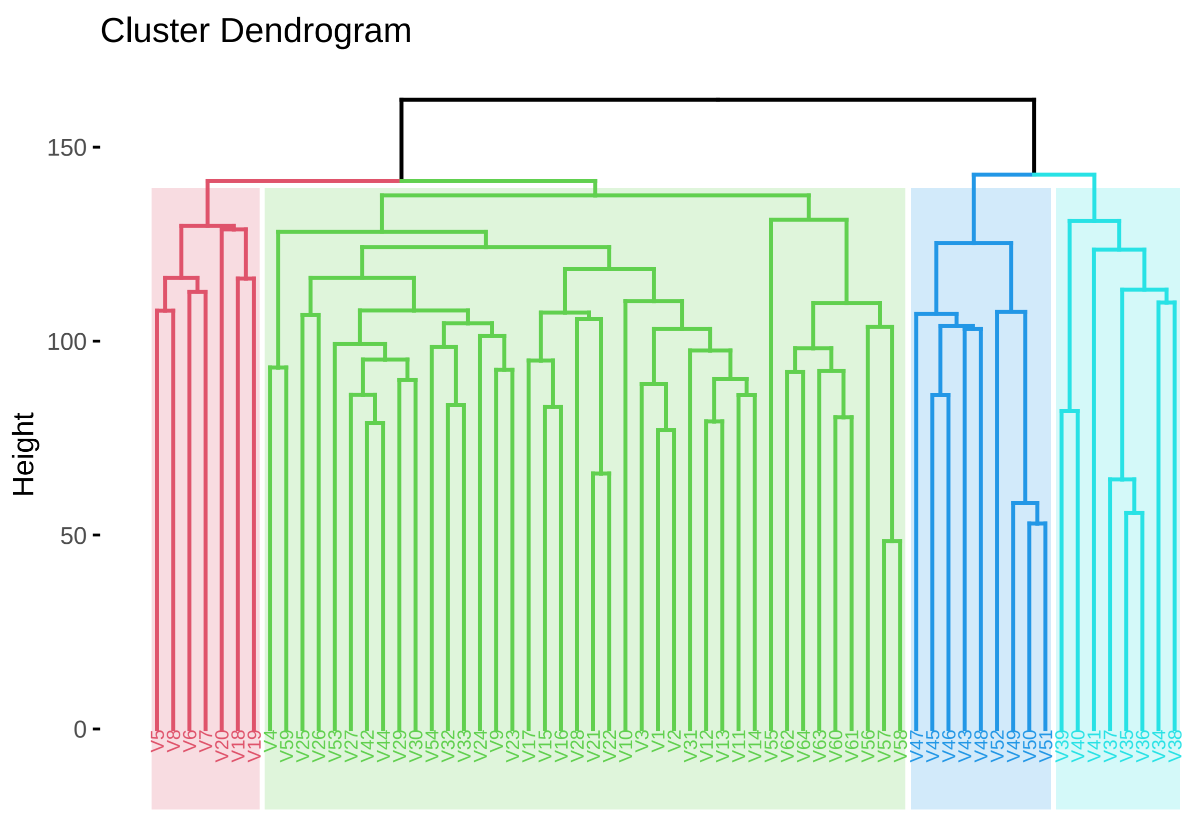
```
## Loading required package: ggplot2
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

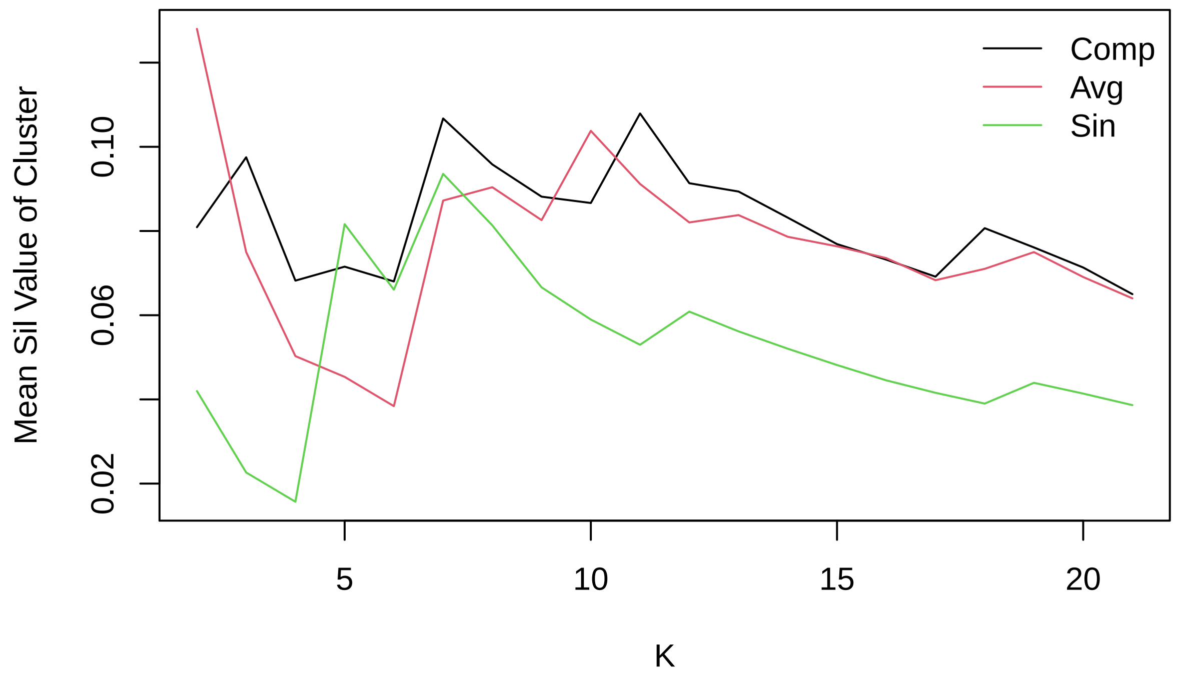


```
### Ignore the warning message

fviz_dend(hc_complete, k = 4, cex = 0.5, k_colors = c(2,3,4,5), color_labels_by_k = TRUE, rect = TRUE,
```

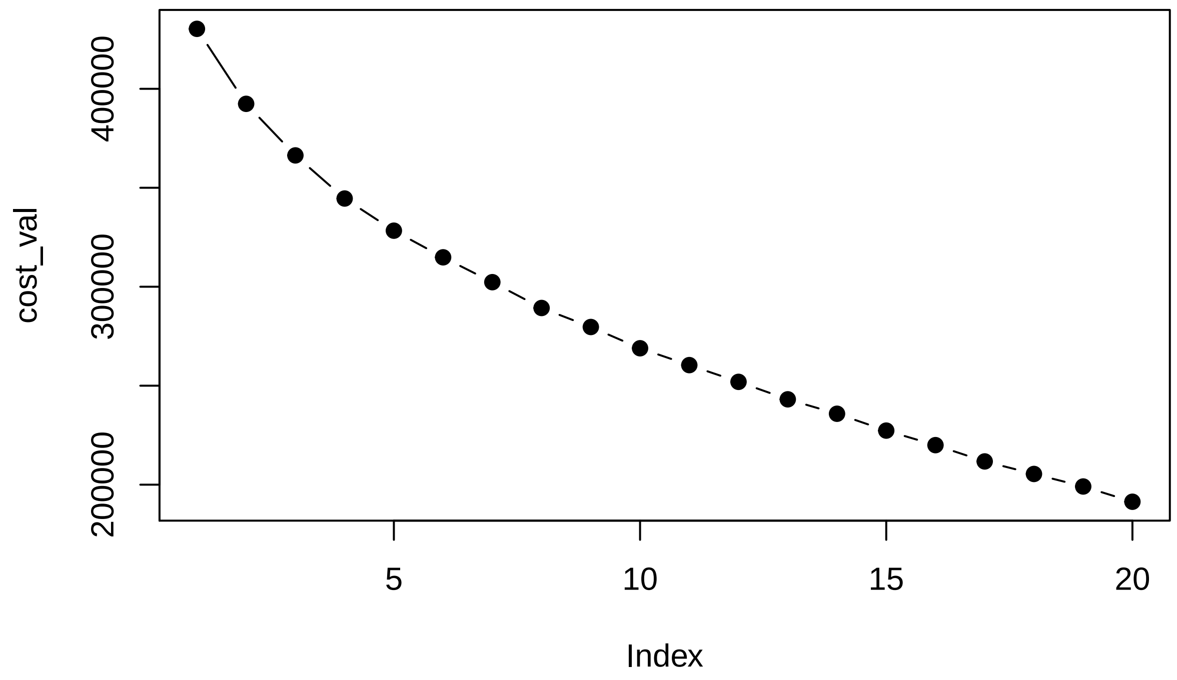
```
## Warning in if (color == "cluster") color <- "default": the condition has length
## > 1 and only the first element will be used
```



```
library(cluster)

set.seed(100)

k       = 1:20
sil_mat = matrix(0, length(k), 4)
colnames(sil_mat) = c("k", "Complete", "Average", "Single")
sil_mat[,1] = k+1

for (i in k)
{
  sil_mat[i,2] = mean(summary(silhouette(x = cutree(hc_complete , k = i+1), dist = dist_x))$clus.avg.wi
  sil_mat[i,3] = mean(summary(silhouette(x = cutree(hc_average ,  k = i+1), dist = dist_x))$clus.avg.wi
  sil_mat[i,4] = mean(summary(silhouette(x = cutree(hc_single ,   k = i+1), dist = dist_x))$clus.avg.wi
}

matplot(x = sil_mat[,1], y = sil_mat[,2:4], ylab ="Mean Sil Value of Cluster" , xlab = "K",
        type = "l", lty = 1, col = c(1,2,3))
legend("topright",c("Comp","Avg","Sin"), lty = 1, col = c(1,2,3), bty = "n")
```



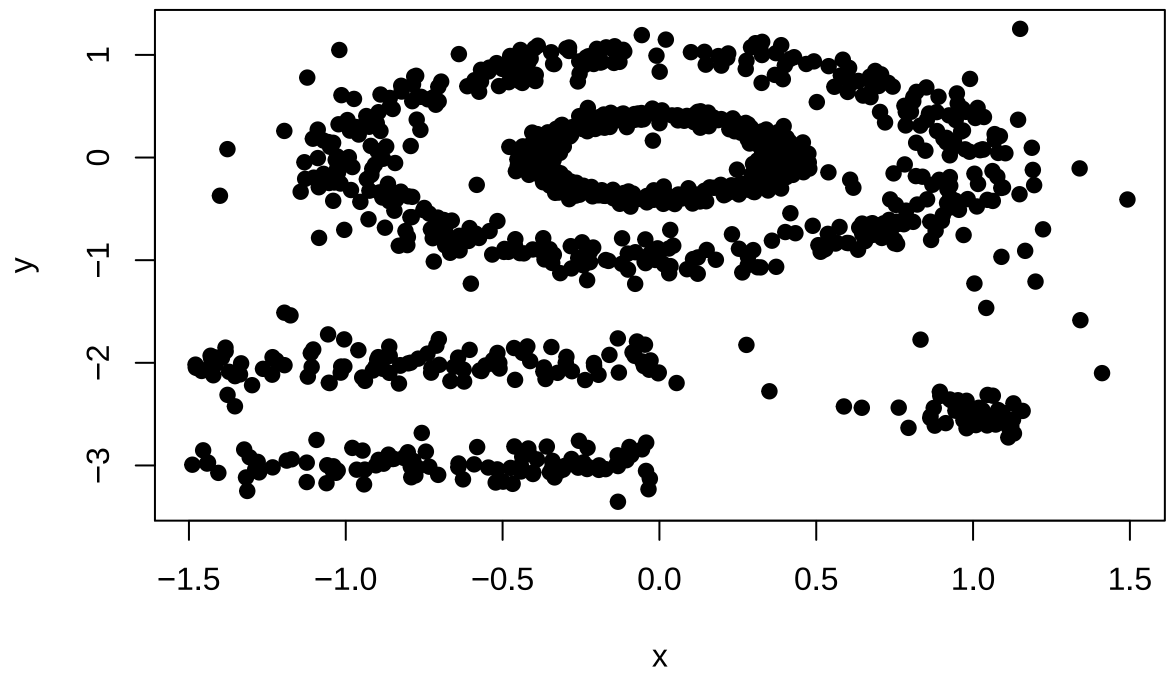
```r
## Try K Means Cluster
m = 20
cost_val = numeric(m)

### This may take 2-3 minutes:

for (i in 1:m)
{
  km_out      = kmeans(x, i, nstart = 20)
  cost_val[i] = km_out$tot.withinss
}

# Elbow plot
plot(cost_val, type = "b", pch = 19)
```



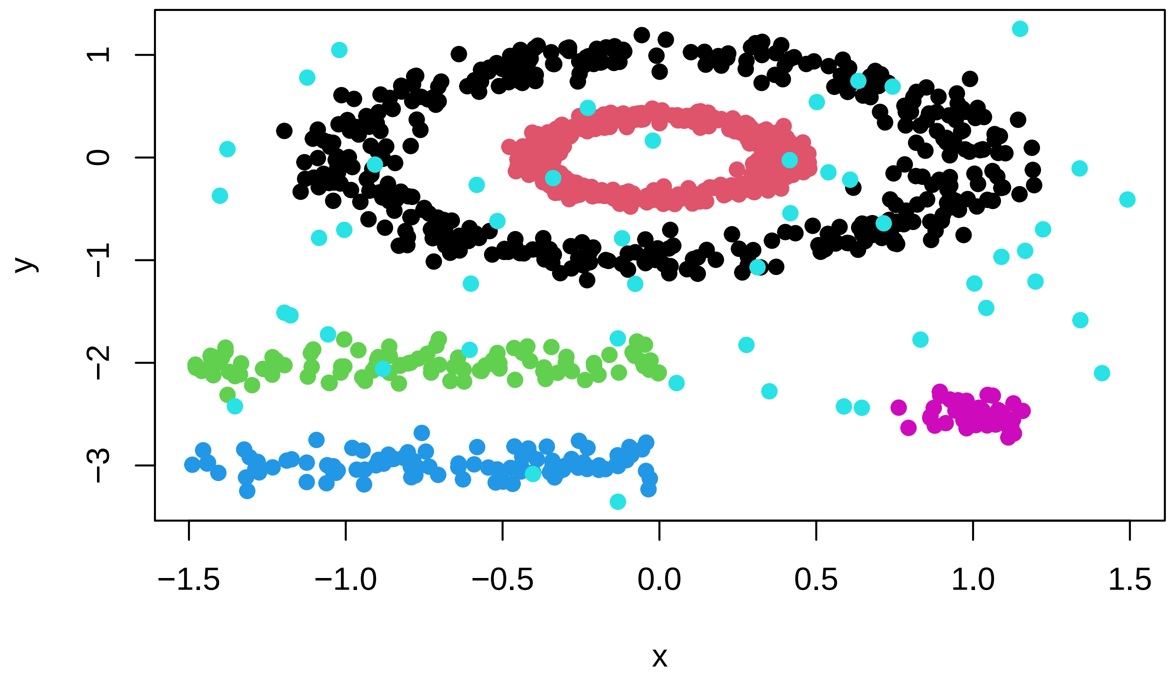
```
################## K Means Limitations and Density Based Clustering#########################

library(factoextra)
data("multishapes")
head(multishapes)
```

```
##            x          y shape
## 1 -0.8037393 -0.8530526     1
## 2  0.8528507  0.3676184     1
## 3  0.9271795 -0.2749024     1
## 4 -0.7526261 -0.5115652     1
## 5  0.7068462  0.8106792     1
## 6  1.0346985  0.3946550     1
```

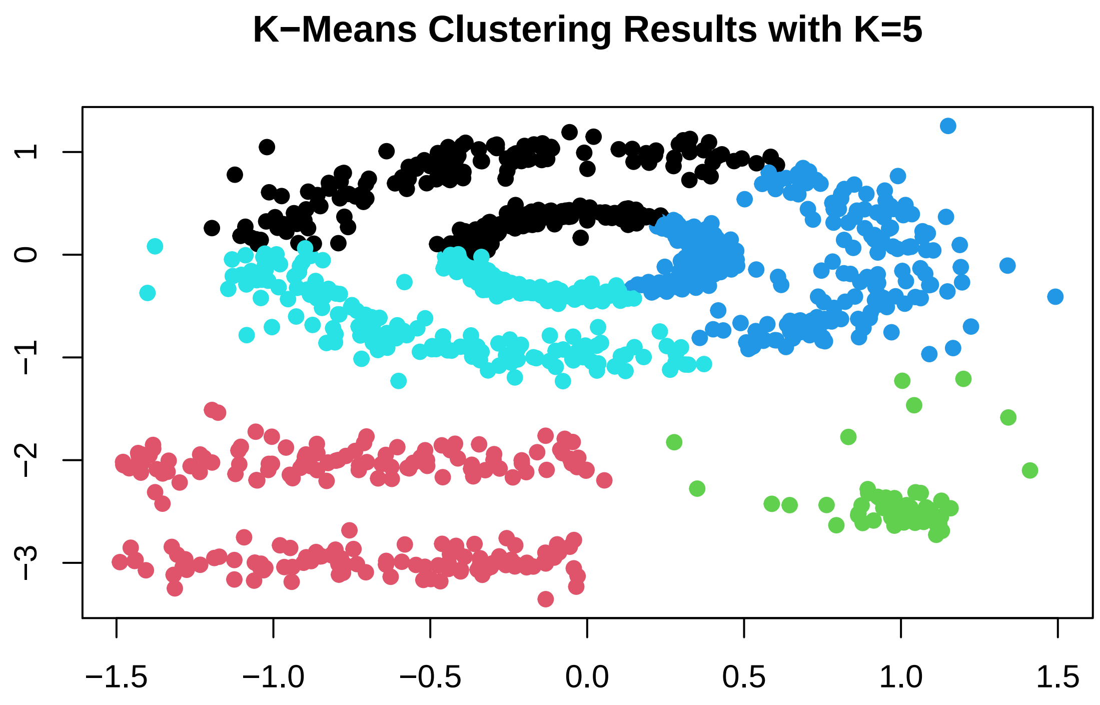
```
### 1:2 correspond to the columns of the data.  Column 3 has the actual labels

x = multishapes[, 1:2]

plot(x, pch = 19)
```



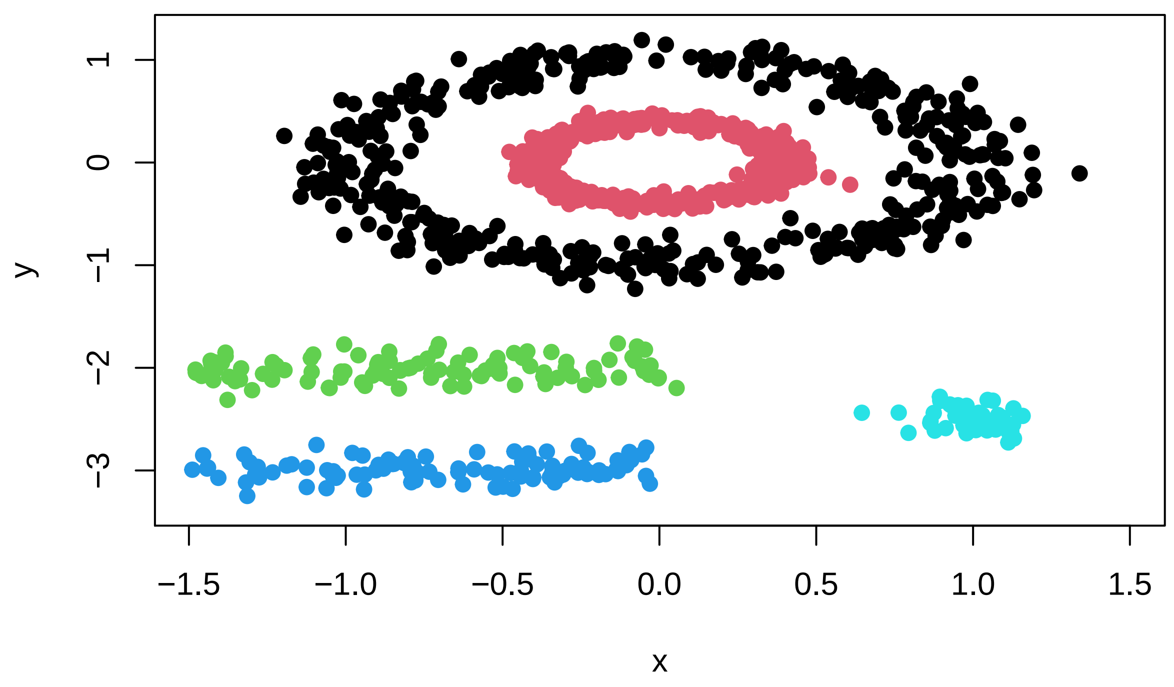
```
plot(x, col = multishapes[, 3], pch = 19)
```



```
set.seed(123456)
km_out = kmeans(x, 5, nstart = 20)

### Have the previous plot up so you can see compare it
###  with the kmeans results below:

plot(x, col =(km_out$cluster) , main="K-Means Clustering Results with K=5", xlab ="", ylab="", pch =19)
```



```
#install.packages("dbscan")
library(dbscan)

### dbscan should be quite fast.
### eps & minPts have *no* defaults.  You must specify them.
### minPts = 5 does *not* mean detect five clusters. It's just a parameter
### of the dbscan that needs to be specified.

db_out = dbscan(x, eps = 0.15, minPts = 5)

plot(x, col=db_out$cluster, pch = 19)
```



```
summary(as.factor(db_out$cluster))
```

```
##   0   1   2   3   4   5
##  31 410 405 104  99  51
```