Assignment 5 Clustering

Naveen Narayanan Meyyappan

07/11/2019

## Kmeans Clustering

Kmeans clustering is a type of unsupervised learning

Let us read the data frame

data = read.csv("kmeans.csv")  
summary(data)

## a b c   
## Min. :-16.299 Min. :-22.9185 Min. :-13.6434   
## 1st Qu.: -8.072 1st Qu.: -5.4970 1st Qu.: -0.2002   
## Median : -1.713 Median : -0.1349 Median : 1.9768   
## Mean : -2.513 Mean : -1.4747 Mean : 2.0669   
## 3rd Qu.: 2.363 3rd Qu.: 3.3220 3rd Qu.: 4.0754   
## Max. : 14.465 Max. : 10.6343 Max. : 14.1380   
## d e   
## Min. :-18.8397 Min. :-12.7962   
## 1st Qu.: -4.6318 1st Qu.: -3.1217   
## Median : 0.6776 Median : -0.8154   
## Mean : 0.1180 Mean : -0.1788   
## 3rd Qu.: 5.2020 3rd Qu.: 2.0573   
## Max. : 14.9545 Max. : 15.9304

The data has 5 numeric columns a,b,c,d and e and 446 records.

Now we have to generate 3 clusters from this data using the kmeans algorithm. Let us define the kclus function which performs the clustering of data.

set.seed(100) #Set the seed so that the same set of random numbers are generated and the result doesnot change  
  
# Kmeans function  
kclus <- function(data, nclus)   
 {  
 # Generating 3 random cluster centers for each column in the data   
 acen <- data[1:nclus,1]  
 bcen <- data[1:nclus,2]  
 ccen <- data[1:nclus,3]  
 dcen <- data[1:nclus,4]  
 ecen <- data[1:nclus,5]  
 #Adding a column to the data frame and renaming the columns  
 data <- data.frame(aval = data[,1], bval = data[,2], cval = data[,3], dval = data[,4], eval = data[,5], clus = NA)  
 #Creating a new dataframe called cluster which maintains the centers of each cluster  
 clus <- data.frame(name = 1:nclus, acen = acen, bcen = bcen, ccen = ccen, dcen = dcen, ecen = ecen)  
   
 # Defining a flag which determines when to stop the algorithm  
 finish <- FALSE  
   
 # Here l stores the dimensions of the data  
 l<-dim(data)  
   
 while(finish == FALSE)   
 {  
 for(i in 1:l[1])   
 {  
 # For each data point calculate the distances between the centers and the datapoints  
 dist <- sqrt((data[i,1]-clus$acen)^2 + (data[i,2]-clus$bcen)^2 + (data[i,3]-clus$ccen)^2 + (data[i,4]-clus$dcen)^2 + (data[i,5]-clus$ecen)^2)  
 # Now assign the data point to one of the nclus clusters.  
 data$clus[i] <- which.min(dist) #The data point is assigned to the cluster for which the distance is minimum among the nclus clusters  
 }  
 #Store the privous centers in a separate column vector  
 acen\_old <- clus$acen  
 bcen\_old <- clus$bcen  
 ccen\_old <- clus$ccen  
 dcen\_old <- clus$dcen  
 ecen\_old <- clus$ecen  
   
 for(i in 1:nclus)   
 {  
 # Calculate the new mean of the clusters and update the cluster data frame  
 clus[i,2] <- mean(subset(data$aval, data$clus == i))  
 clus[i,3] <- mean(subset(data$bval, data$clus == i))  
 clus[i,4] <- mean(subset(data$cval, data$clus == i))  
 clus[i,5] <- mean(subset(data$dval, data$clus == i))  
 clus[i,6] <- mean(subset(data$eval, data$clus == i))  
 }  
 # We can stop this process when there is no more change in the cluster centers  
 if(identical(acen\_old, clus$acen) & identical(bcen\_old, clus$bcen) & identical(ccen\_old, clus$ccen) & identical(dcen\_old, clus$dcen) & identical(ecen\_old, clus$ecen))  
 {  
 finish <- TRUE # End the loop by changing the flag  
 }  
 }  
 output<-NULL  
 output$data<-data  
 output$cluster<-clus  
 return(output) #Returning both the data and cluster data frames   
}

Let us call the kclus function on our data set

output <- kclus(data, 3)   
# The above ouput variable has both the data and cluster centers  
  
#Let us have a look at the output data frame  
head(output$data) # Output is a data frame with an additional clus column. This column indicates the cluster to which the data point belongs

## aval bval cval dval eval clus  
## 1 0.2021097 3.810452 2.807222 9.693731 -5.28247675 3  
## 2 1.0283248 4.909634 1.997119 8.085414 -1.65859459 3  
## 3 0.8123358 7.513041 1.705482 10.948794 -1.91342288 3  
## 4 -1.1963861 4.861313 1.020611 8.315357 -1.16201757 3  
## 5 3.1835864 6.354218 3.299007 6.610393 -1.60941745 3  
## 6 0.8543143 5.896485 1.148101 7.418075 0.05425219 3

table(output$data[,6]) # The output below indicates the number of data point that belongs to the various clusters

##   
## 1 2 3   
## 150 189 107

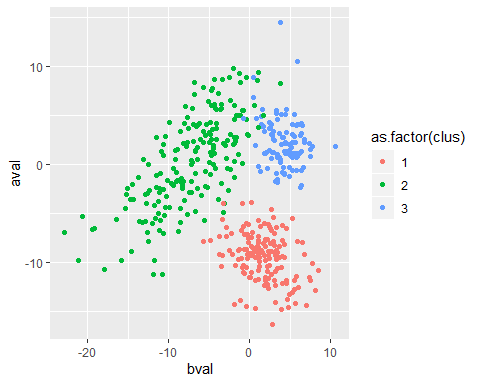
# Now let us look at the covariance of the cluster matrix  
print(cov(output$cluster))

## name acen bcen ccen dcen ecen  
## name 1.0000000 5.7972894 1.2789064 1.4495542 3.297378 -0.8035381  
## acen 5.7972894 38.7768394 -5.9114497 9.1720994 4.307647 0.8541706  
## bcen 1.2789064 -5.9114497 35.9938123 -0.1279143 42.397833 -15.2408579  
## ccen 1.4495542 9.1720994 -0.1279143 2.2155140 2.577486 -0.3449638  
## dcen 3.2973782 4.3076475 42.3978327 2.5774860 53.301370 -18.4440926  
## ecen -0.8035381 0.8541706 -15.2408579 -0.3449638 -18.444093 6.5253533

The output is plotted as shown below:

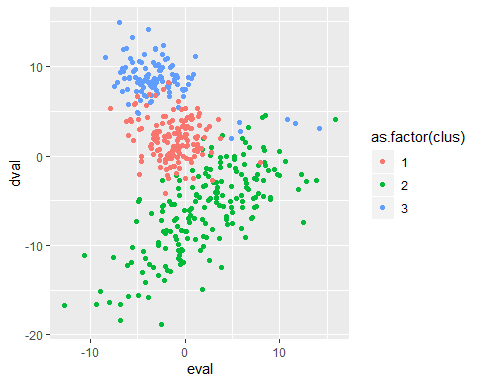
The following is a plot between the 2 columns from the data (“a” and “b”). The clusters are plotted with different colors

ggplot(output$data, aes(bval, aval, color = as.factor(clus))) + geom\_point()



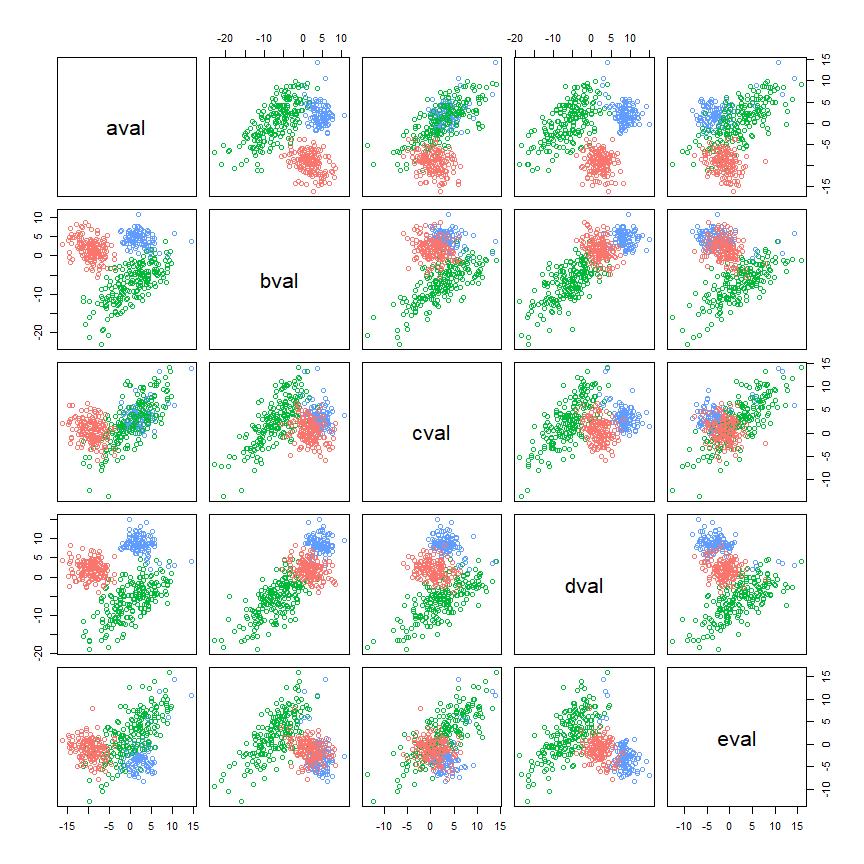
Similarly, we can call other plots by changing the values on the 2 axes. Another plot would be between columns “d” and “e”

ggplot(output$data, aes(eval, dval, color = as.factor(clus))) + geom\_point()



Now we can use the inbuilt pairs plot to get a view of all the plots between the different columns

X <- output$data[,1:5]  
y <- output$data[,6]  
y\_col <- c('#F8766D', '#00BA38', '#619CFF')  
  
pairs(X, col = y\_col[y])



### References

1. <https://en.wikipedia.org/wiki/K-means_clustering>
2. <https://stackoverflow.com/questions/41912875/writing-own-kmeans-algorithm-in-r>