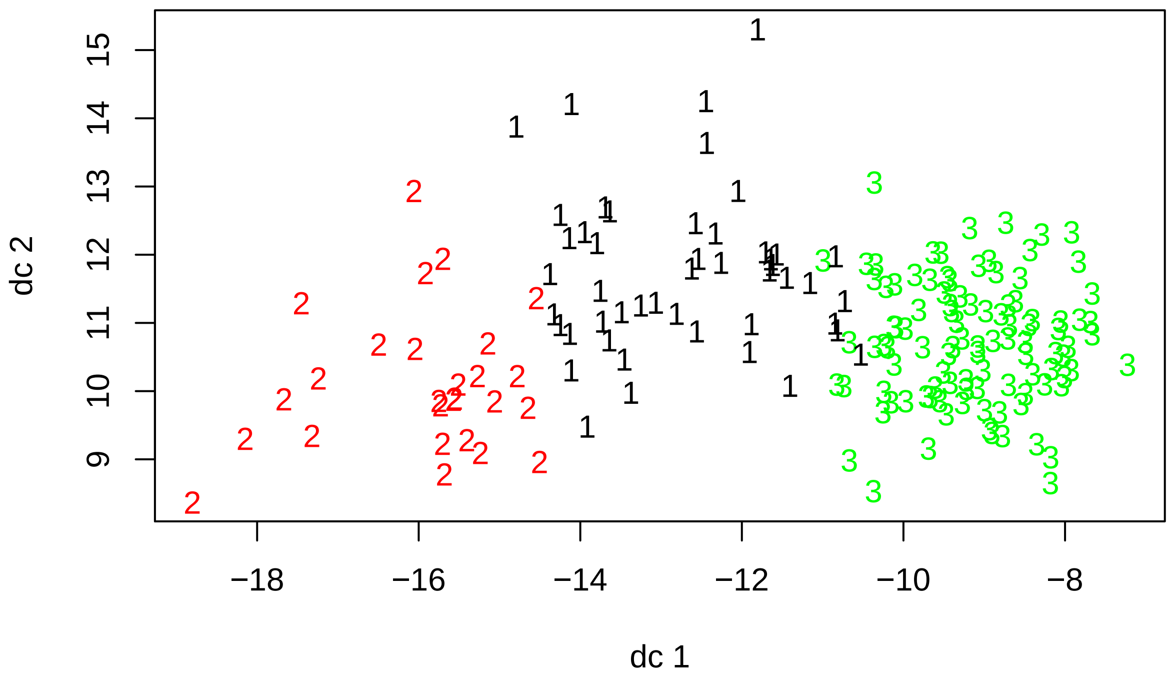
STAT 37810 Assignment 4 K-means

R implementaiton of K-means clustering algorithm

```r
##--- L2distance ------------------------------------------##
# find the L2 distance between two vectors
# vec1 : first vector
# vec2 : second vector
L2distance <- function(vec1, vec2) {
  sqrt(sum(vec1-vec2)^2)
}


##--- assign cluster labels --------------------------------##
# assign each data entry of the input matrix "mat" to one of the "k" clusters
# according to the centroid (mean)
# mat : input matrix
# centroids : centroids/mean for labeling
# k : number of clusters
# adj : paramter for column index matching
assign <- function(mat, centroids, k, adj = 1){
  dist <- matrix(0,nrow=1,ncol=k)  # distance matrix
  r <- dim(mat)[1]; c<-dim(mat)[2];   # dimensions
  label <- numeric(r)
  for (i in c(1:r)) {
    for (j in c(1:k)) {
      dist[1,j] <- L2distance(mat[i,1:c-adj],centroids[j,])
      # compute the distance bewteen each r data entry and k means
    }
    label[i] <- which.min(dist)
    # assign to cluster of the mean with which
    # the distance between the entry and the mean is minimum
  }
  return(label)
}


##--- recalculate centroids --------------------------------##
# recalculate the mean given the updated cluster labeling for each cluster
# mat: input matrix
# centernumber : numbering of cluster (1 ~ k)
recenter <- function(mat,centernumber){
  r <- dim(mat)[1]; c <- dim(mat)[2] # dimensions
  # for loop version
  #  count = 0; newcenter <- numeric(c-1)
  #   for (i in c(1:r)) {
  #     if (mat[i,c] == centernumber) {
  #       newcenter <- newcenter + as.matrix(mat[i,(1:c-1)])
  #       dimnames(newcenter) <- NULL
  #       count = count + 1
  #     }
  #   }
  # using apply
  index <- mat[,c] == centernumber
  # subset the data entries that belong to each cluster
  return( apply(mat[index,(1:c-1)],2,sum) / sum(index) )
  # find the mean of each cluster
  # apply "sum" function over "mat[index,(1:c-1)]" (only featuers)
  # "2" (column) wise
}


##---------- main kmeans method (dowhileloop) ----------------------------##
# mat: input matrix
# k : number of cluters
kmeans <- function(mat,k) {
  r <- dim(mat)[1]; c <- dim(mat)[2] #dimensions of the matrix
  adj <- 0 # for column number matching

  maxmin <- as.numeric(as.vector(apply(mat,2,max)[1:c])) -
            as.numeric(as.vector(apply(mat,2,min)[1:c]))
  threshold <- 10^(4 - ceiling(log(sum(maxmin), base=10)))
  # my own threshold idk okay?
  # negative power of 10 based on the order of magnitude of min max difference and

  # initial centroid (uniform random)
  max <- apply(mat,2,max); min <- apply(mat,2,min)
  # max and min of the features
  centroids <- mapply(runif, k, min = min, max = max)
  # generate k random vectors of length = number of features
  # from uniform distribution of min and max of each features

  # distance matrix for updating cluster label
  dist <- matrix(0,nrow=1,ncol=k)

  # initial assignment of the clusters
  # (seperate b.c. restricting column numbers on L2 distance calculation later steps)
  mat[,c+1] <- assign(mat,centroids,k,adj=0)

  repeat {
    #update cluster label based on new centroids (1+i iterations)
    mat[,c+1] <- assign(mat,centroids,k,adj=1)

    temp <- centroids # for steady state check

    #update centroids
    for (i in c(1:k)) {
      centroids[i,] <- recenter(mat,i)
    }

    change <- sum((temp - centroids)^2)
    # find the change in mean of each clusters

    # print(centroids) # testing
    if(change <= threshold) break
  }
  return(mat[,c+1])
}
##---------------------------------------------------------------------------##
```

Testing kmeans on “wine” data by “fpc” cluster graphics package.

```
require(fpc); data(wine, package="rattle"); attach(wine)
wine <- wine[,2:14] # exclude the categorical variable
wine[,14]<-kmeans(wine,3)
colnames(wine)[14] <- "cluster"
par(mfrow=c(1,1))
plotcluster(wine[1:13], wine$cluster)
```


dc 2
dc 1