STAT 37810 Assignment 4 K-means

R implementaiton of K-means clustering algorithm

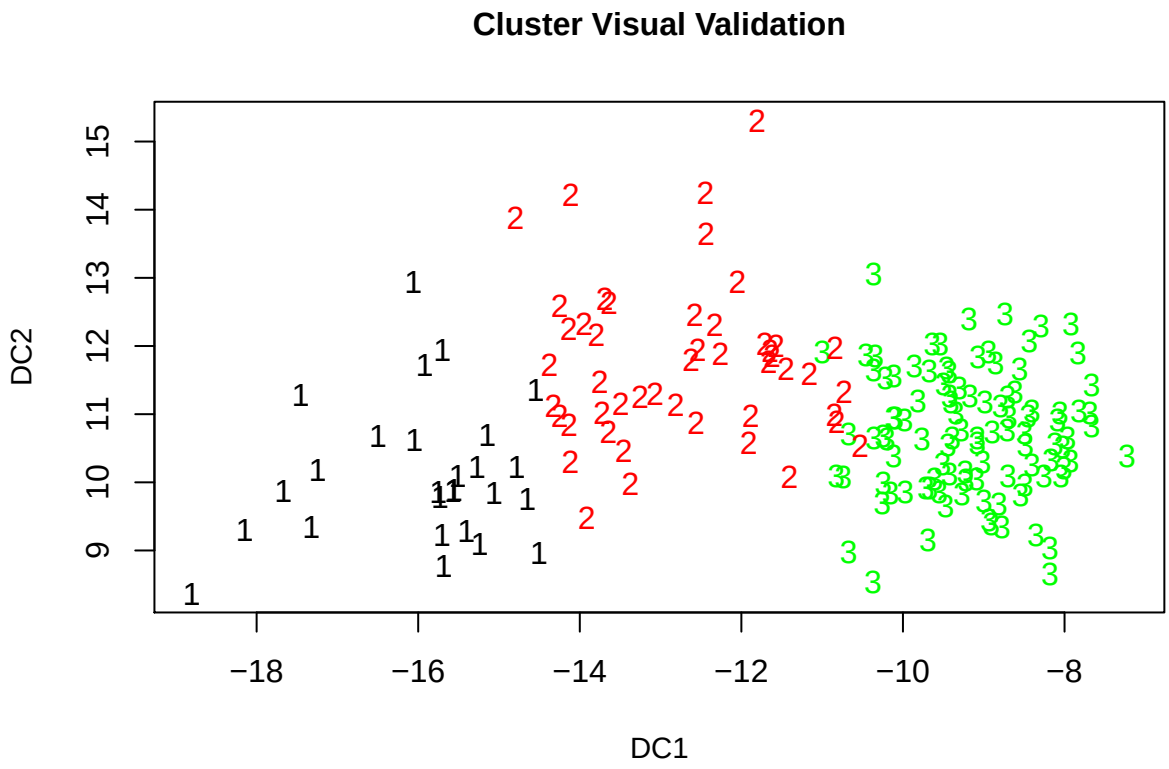
```
##--- L2distance -----------------------------------------##
# find the L2 distance between two vectors
# vec1 : first vector
# vec2 : second vector
L2distance <- function(vec1, vec2) {
  sqrt(sum(vec1-vec2)^2)
}


##--- 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 kmeansR method (dowhileloop) ----------------------------##
# KmeansR to avoid overwritting built in R kmeans function
# mat: input matrix
# k : number of cluters
kmeansR <- 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 kmeansR on "wine" data by "fpc" cluster graphics package.

```
require(fpc); data(wine, package="rattle"); attach(wine)
#set.seed(19920922)
type <- wine[,1]
wine <- wine[,2:14] # exclude the categorical variable
wine[,14]<-kmeansR(wine,3)
colnames(wine)[14] <- "cluster"
par(mfrow=c(1,1))
plotcluster(wine[1:13], wine$cluster, main="Cluster Visual Validation", xlab="DC1", ylab="DC2",
            cex.main=1.0, cex.lab=0.9)
```



The fpc cluster plot shows the DC1 and DC2 on x and y axis, respectively. Consider the dimensionality reduction of the wine data (13 continuous features) by using linear combinations. Such process is called Linear Discriminant Analysis (LDA) and returns the discriminant coordinates which are linear combinations of existing features that best explains the data. The dc1 and dc2 are 1st and 2nd discriminant coordinates.

The fpc cluster plot suggest the kmeansR does cluster the wine data pretty well. Plots will look different because kmeansR start with randomly selelcted initial centroids. Notice points that are closed to one another in DC1-DC2 space are clustered togheter.

Provided that we already know the "answer" to the wine data classification problem. We may use the exisiting labels to see how well our Kmeans clustering algorithsm cluster the data. However, in theory, clustering is un-supervised learning and Kmeans does not "classify" data. Anyways making use of our intial labels.

```
labs <- wine[,14]
labs <- cbind(labs,as.matrix(type))
colnames(labs)[1]<-"cluster"; colnames(labs)[2]<-"type"
as.factor(labs[2,]);as.factor(labs[1,])
```

```
## cluster    type
##       2       1
## Levels: 1 2
```

```
## cluster    type
##       2       1
## Levels: 1 2
```

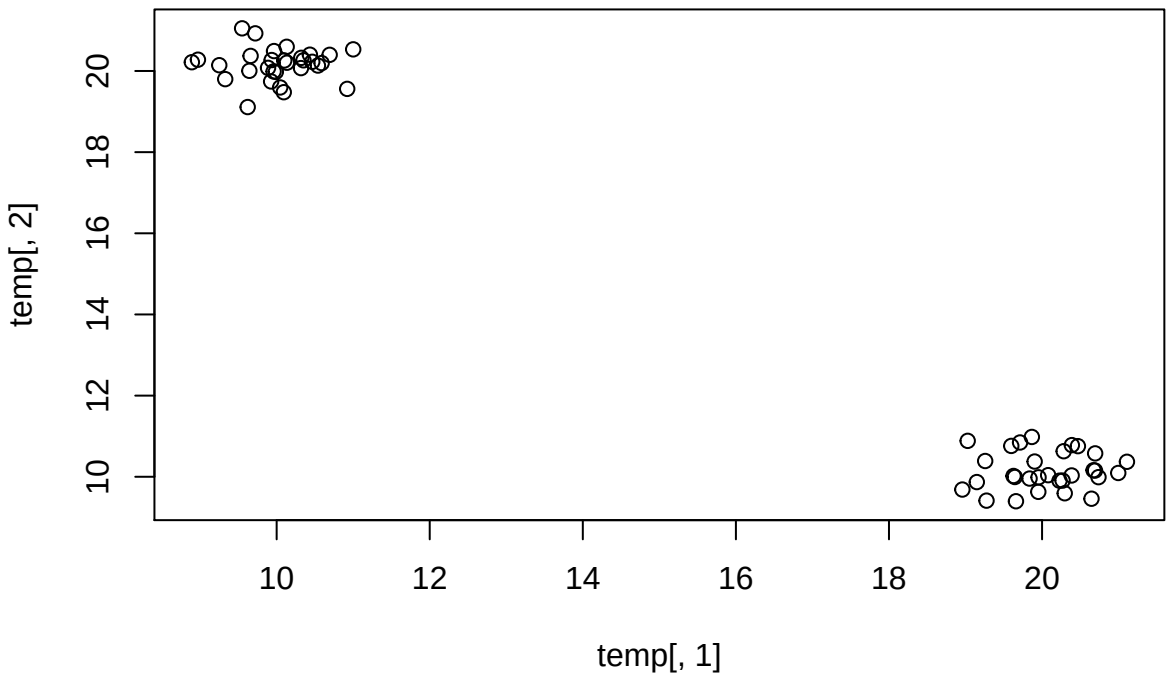
```
#(labs[,1]==2)
#(labs[,2]==3)

#hmmm
```

Another graphical tool for visualizing cluster validity is using distance matrix heatmap. Which turns the distance matrix into the color coded diagram. A large distance is darker while close distance is bright. Ideally, you want points that are closed to be clustered together so the diagonal sections are darker. Consider the following example.

```
require(RColorBrewer);require(pheatmap)
```

```
## Loading required package: RColorBrewer
## Loading required package: pheatmap
```

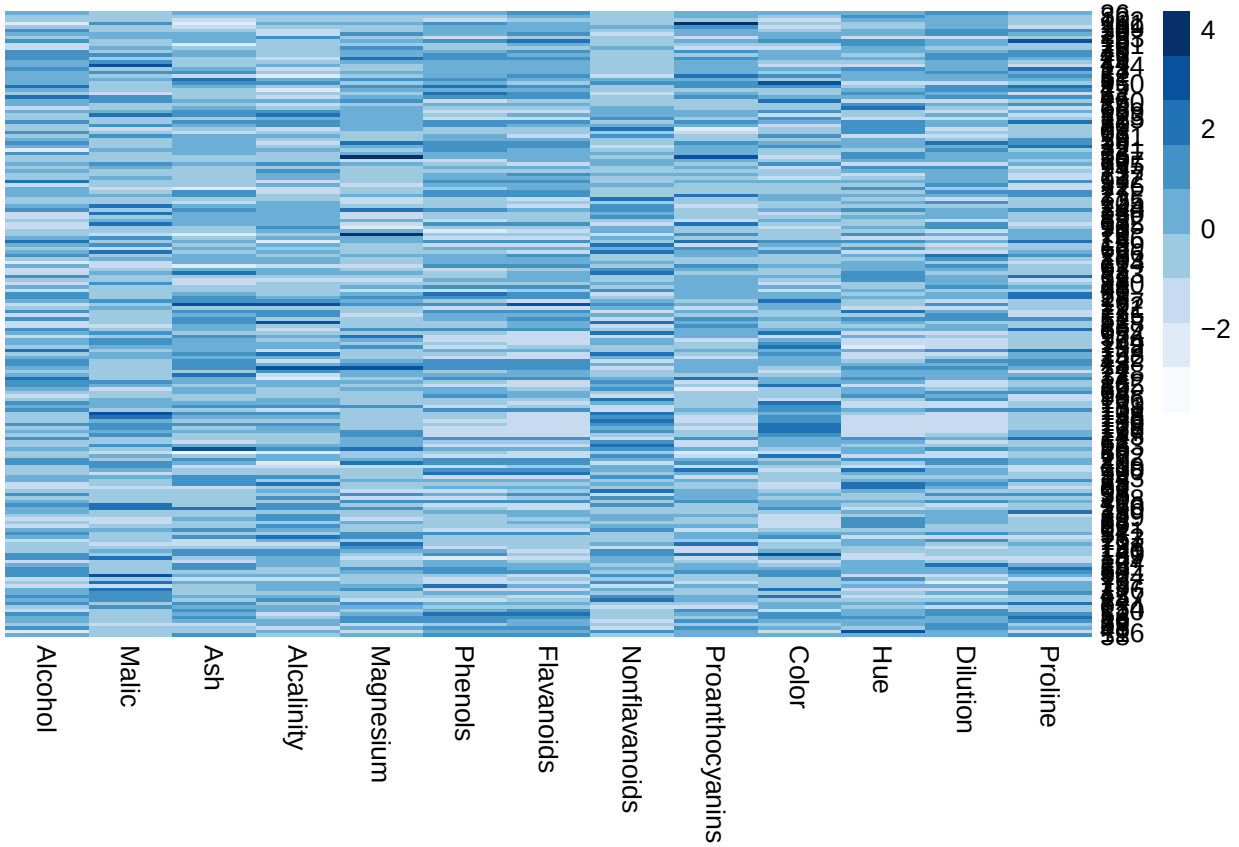
```
temp <- matrix(0,ncol=2,nrow=60)
temp[1:30,1] <- rnorm(30,10,0.5); temp[31:60,1] <- rnorm(30,20,0.5)
temp[1:30,2] <- rnorm(30,20,0.5); temp[31:60,2] <- rnorm(30,10,0.5)
plot(temp[,1],temp[,2])
```



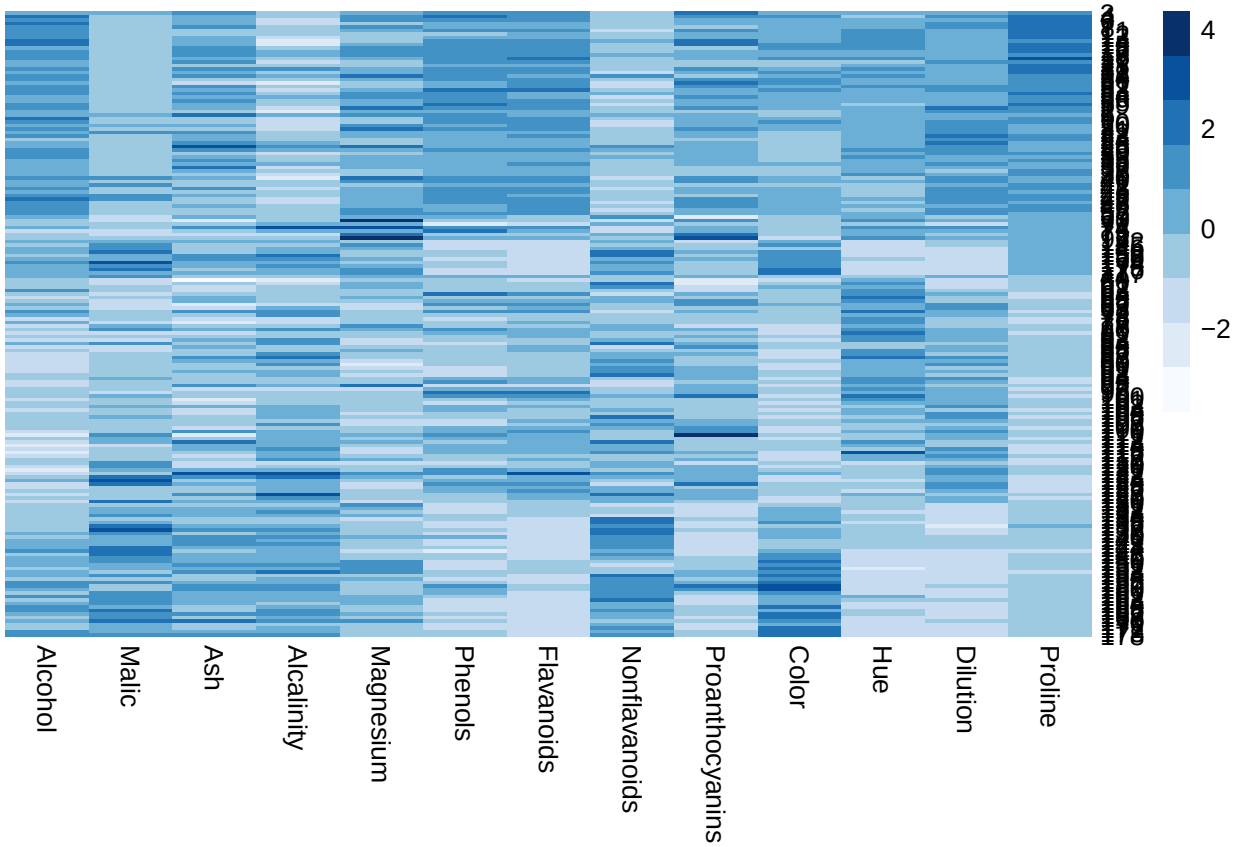

```
pheatmap(scale(temp),cluster_rows=F,cluster_cols=F, col=brewer.pal(9, "Blues")[1:9], border_color=NA)
```

However, it turns out heatmap is not an ideal tool for wine data which appears to be pre "clustered" by the type. Mixing the wine data by using the index of sample(178) does seem to suggest closes points are clustered together but it is not as clear as fpc cluster plot

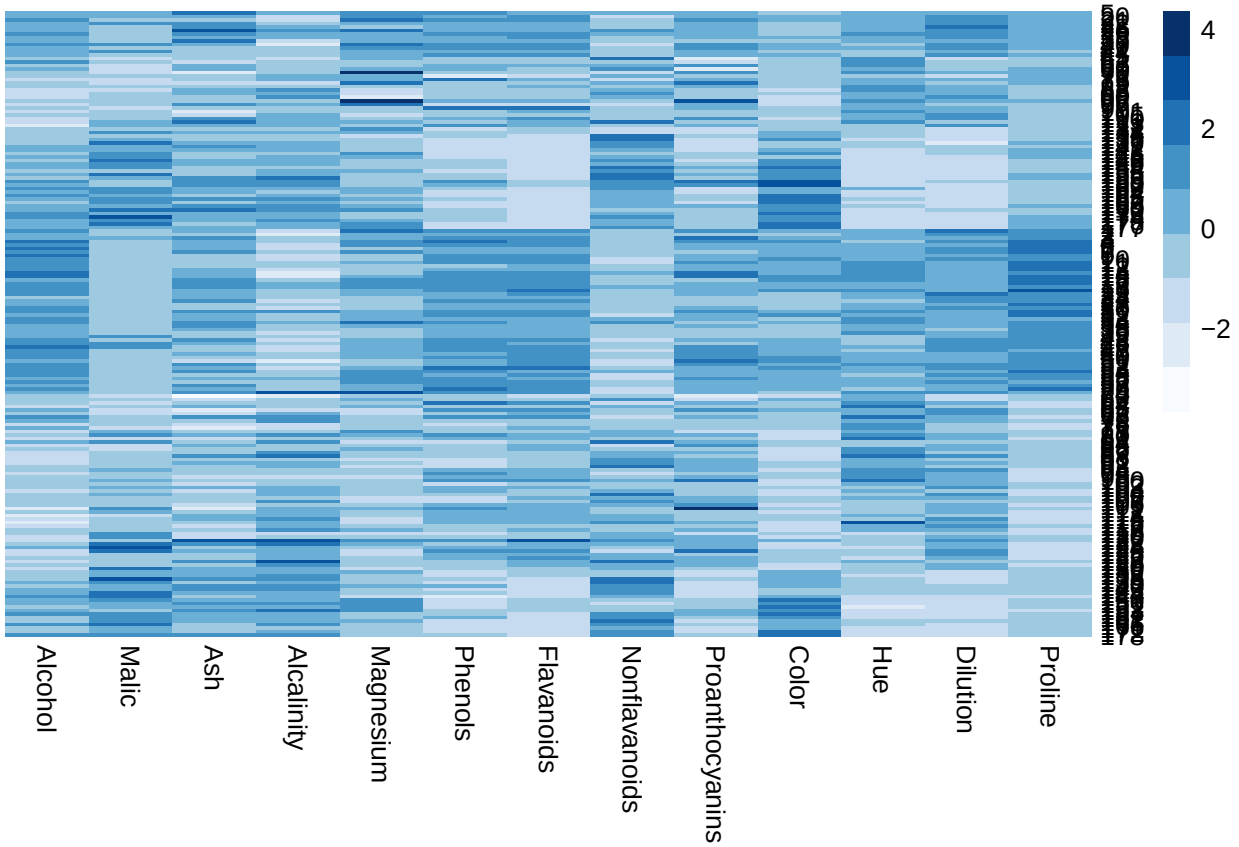
```
wine.mixed <- wine[sample(178),]
pheatmap(scale(wine.mixed[,1:13]), cluster_rows=F,cluster_cols=F, col=brewer.pal(9, "Blues")[1:9], bord
```




```
wine.orderd <- wine[order(wine$cluster),]
pheatmap(scale(wine.orderd[,1:13]), cluster_rows=F,cluster_cols=F, col=brewer.pal(9, "Blues")[1:9], bor
```




```
wine[,15] <- kmeans(wine,3)$cluster
wine.orderd2 <- wine[order(wine$V15),]
pheatmap(scale(wine.orderd2[,1:13]), cluster_rows=F,cluster_cols=F, col=brewer.pal(9, "Blues")[1:9], bo
```


4
2
0
−2
Alcohol
Malic
Ash
Alcalinity
Magnesium
Phenols
Flavanoids
Nonflavanoids
Proanthocyanins
Color
Hue
Dilution
Proline