

K-Means

Seung Ah Ha, Jaymo Kim, Wonbin Song

Algorithm: First randomly assign a group label from 1 to 3 to each observations. Then calculate the current group centers and re-assign each observation to the closest cluster using the Euclidean distance. We iterate this procedure until the assignment of observations to groups stops changing.

We implemented the algorithm by first creating the function “initialft” that calculates the group centers (which can be calculated by mean) and the Euclidean distances between each data points and those group centers. Then, it is included to the new group that the Euclidean distance to group center is shortest. Finally, we created the function called “kmeans_ft” that iterates the function “initialft” to reassign data points into the group that is closest to them until there is no more change to their group label.

Wine data

```
# install.packages('rattle')
data(wine, package="rattle")

initialft <- function(mat.data, n.row, n.col, G.label, new.data) {
  # calculates the group centres
  mat.mu <- matrix(0,3,(n.col+1))
  for(j in 2:(n.col+1)){
    for(i in 1:3){
      mat.mu[i,j] <- mean(new.data[G.label==i,j])
    }
  }
  mat.mu[1,1]<- 1
  mat.mu[2,1]<- 2
  mat.mu[3,1]<- 3

  # calculates the Euclidean distance
  distance <- matrix(0,n.row,3)
  new.label <- rep(0,n.row)
  for(i in 1:n.row){
    distance[i,1] <- sqrt(sum((new.data[i,]-mat.mu[1,])^2))
    distance[i,2] <- sqrt(sum((new.data[i,]-mat.mu[2,])^2))
    distance[i,3] <- sqrt(sum((new.data[i,]-mat.mu[3,])^2))
  }

  # re-assign
  for (i in 1:n.row){
    new.label[i] <- which(distance[i,]==min(distance[i,]))
  }

  return(new.label)
}

kmeans_ft <- function(data)
{
  mat.data <- data.matrix(data)
```

```

n.row <- nrow(mat.data)
n.col <- ncol(mat.data)

# randomly assigning a group label
G.label <- rep(0,n.row)
for(i in 1:n.row){
  G.label[i] <- sample(1:3,1)
}
new.data <- cbind(G.label,mat.data)

new.l <- initialft(mat.data, n.row, n.col, G.label, new.data)
n.iter <- 1
while(any(new.l != G.label))
{
  G.label <- new.l
  new.l <- initialft(mat.data, n.row, n.col, G.label, new.data)
  n.iter <- n.iter + 1
}
return(list(number_of_iter = n.iter, new_group = new.l))
}

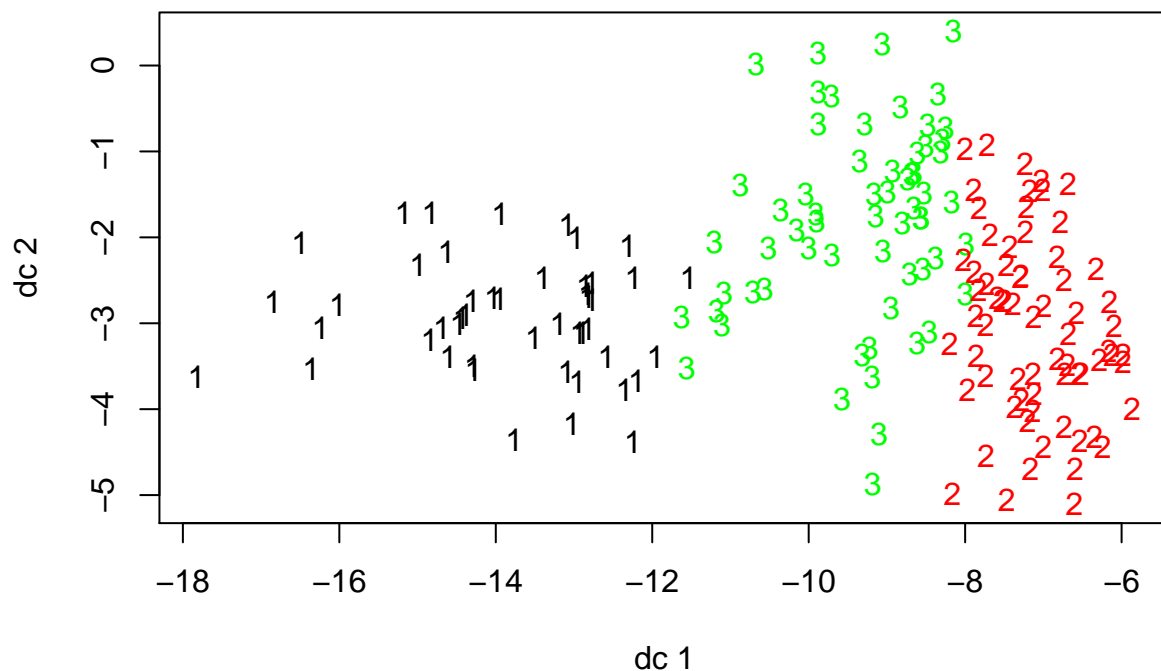
```

Then, we can plot how the groups have been decided. As we can see in the plot, the group 1 is located on the left side and the group 2 is located on the middle while the group 3 is located on the right. That is, we can tell that k-means procedure for this “wine” data is successful from the plot.

```

# install.packages("cluster")
# install.packages("fpc")
library(cluster)
library(fpc)
set.seed(5)
cluster <- kmeans_ft(wine[-1])
plotcluster(wine[-1], cluster$new_group)

```



```

cluster

## $number_of_iter
## [1] 6
##
## $new_group
## [1] 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 1 1 3 3 1 1 3 1 1 1 1 1
## [36] 3 3 1 1 3 3 1 1 3 3 1 1 1 1 1 1 1 1 1 1 1 1 2 3 2 3 2 2 3 2 2 3 3
## [71] 3 2 2 1 3 2 2 2 3 2 2 3 3 2 2 2 2 2 3 3 2 2 2 2 3 3 2 3 2 3 2 2 2 3
## [106] 2 2 2 2 3 2 2 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 3 3 3 3 2 2 2
## [141] 3 3 2 2 3 3 2 3 3 2 2 2 2 3 3 3 2 3 3 2 3 2 3 3 2 3 3 3 2 2 3 3 3
## [176] 3 3 2

type.g1 <- wine[,1][which(cluster$new_group==1)]
accurate1 <- length(type.g1[type.g1==1])/length(type.g1)
type.g2 <- wine[,1][which(cluster$new_group==2)]
accurate2 <- length(type.g1[type.g2==2])/length(type.g2)
type.g3 <- wine[,1][which(cluster$new_group==3)]
accurate3 <- length(type.g3[type.g3==3])/length(type.g3)
accurate1; accurate2; accurate3

## [1] 0.9787234
## [1] 0.7246377
## [1] 0.4677419

```

However, if we compare the result of new group label after k-means with the original ‘Type’ in the wine data set, we can see some discrepancies. For group 1, it matches well to the ‘Type’ in the data, while for group 2 and group 3, there are discrepancies between Type and groups. For group 2, we can see the mixture of Type 1, 2 and 3, and for group 3, it is consisted of Type 2 and Type 3.

Scaled Wine data

Now, we use the scaled data of the wine data. The “scale” function calculates each column’s mean and standard deviation, then scales each elements by subtracting the mean and then dividing by the standard deviation.

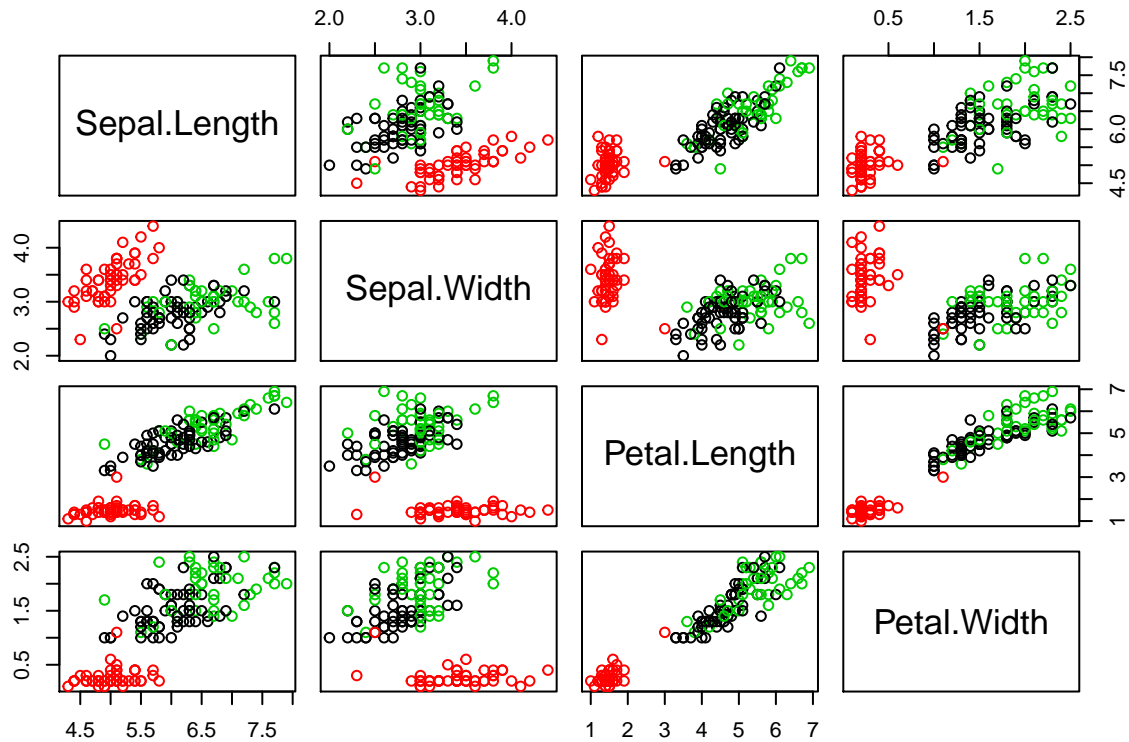
```

set.seed(6)
data.train <- scale(wine[-1])
cluster.scale <- kmeans_ft(data.train)
plotcluster(data.train, cluster.scale$new_group)

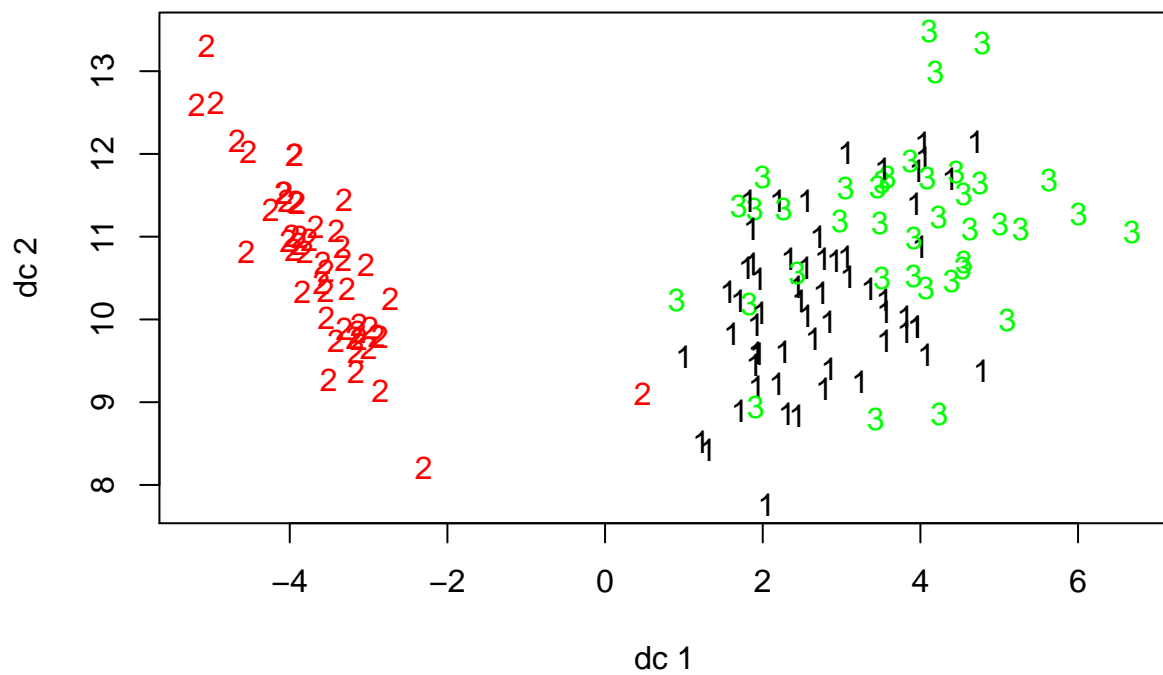
```


Iris data

```
set.seed(10)
cluster2 <- kmeans_ft(iris[, -5])
with(iris, pairs(iris[-5], col=c(1:3)[cluster2$new_group]))
```



```
plotcluster(iris[-5], cluster2$new_group)
```



```

k <- kmeans_ft(iris[,1:4])
k

## $number_of_iter
## [1] 5
##
## $new_group
## [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
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 3 1 3 1 3 3 3 1 3 3 3 3 3 1 3 3 3 3 1
## [71] 1 3 3 3 1 1 3 3 3 3 3 3 3 1 3 1 3 3 3 3 1 3 3 2 3 3 1 3 3 3 1 3 1 1 1
## [106] 3 3 1 1 3 3 1 3 3 1 1 1 1 1 3 1 3 3 1 3 3 3 3 3 1 1 3 1 3 1 3 3 1 3 3
## [141] 3 3 3 3 1 1 3 1 1 1

d1 <- iris[,5][which(k$new_group==1)]
accu1 <- length(d1[d1=="versicolor"])/length(d1)
d2 <- iris[,5][which(k$new_group==2)]
accu2 <- length(d2[d2=="setosa"])/length(d2)
d3 <- iris[,5][which(k$new_group==3)]
accu3 <- length(d3[d3=="virginica"])/length(d3)
accu1; accu2; accu3

## [1] 0.3513514
## [1] 0.9803922
## [1] 0.4193548

```

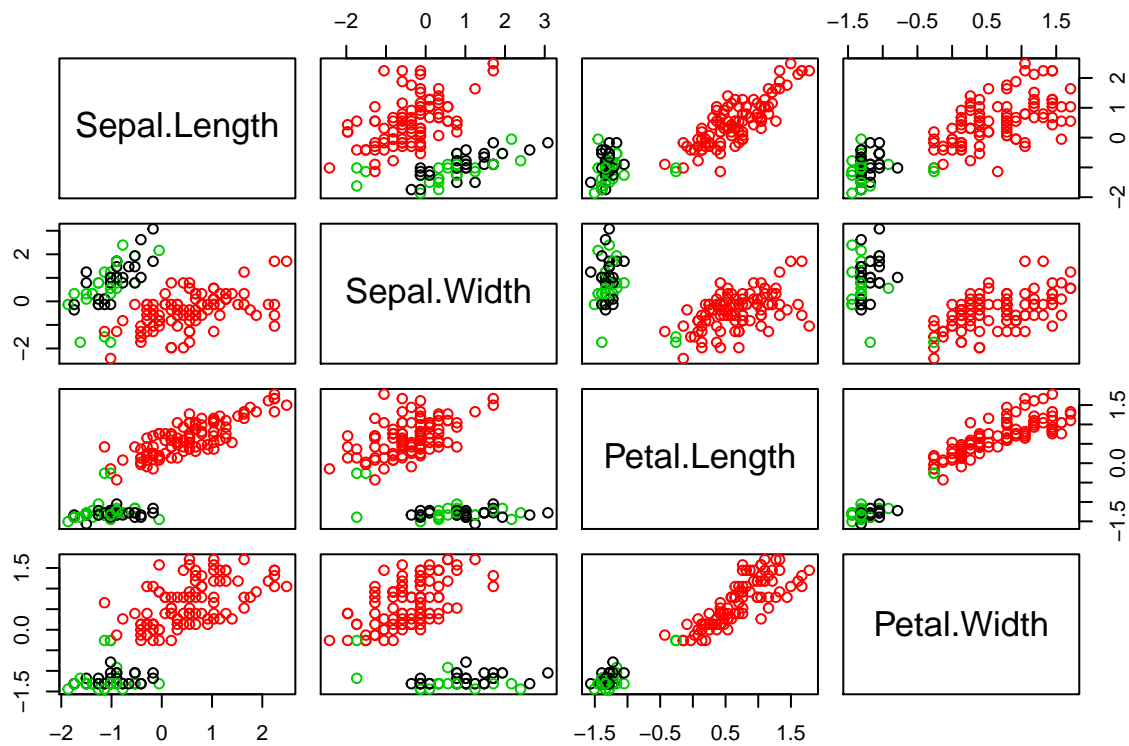
Now, let's look at the iris data set. As we can see in the plot, the data set is only well separated for one type of species. However, the other two species are clustered together and are hard to distinguish. In addition, if we calculate the accuracy rate of the result from k-means to the Species in the original iris data set, we can check that the accuracy rates for one of the three groups is almost 1 while the other two groups are below 0.5, indicating that our k-means function has only successfully separated the data set into 2 groups, not 3.

Scaled Iris data

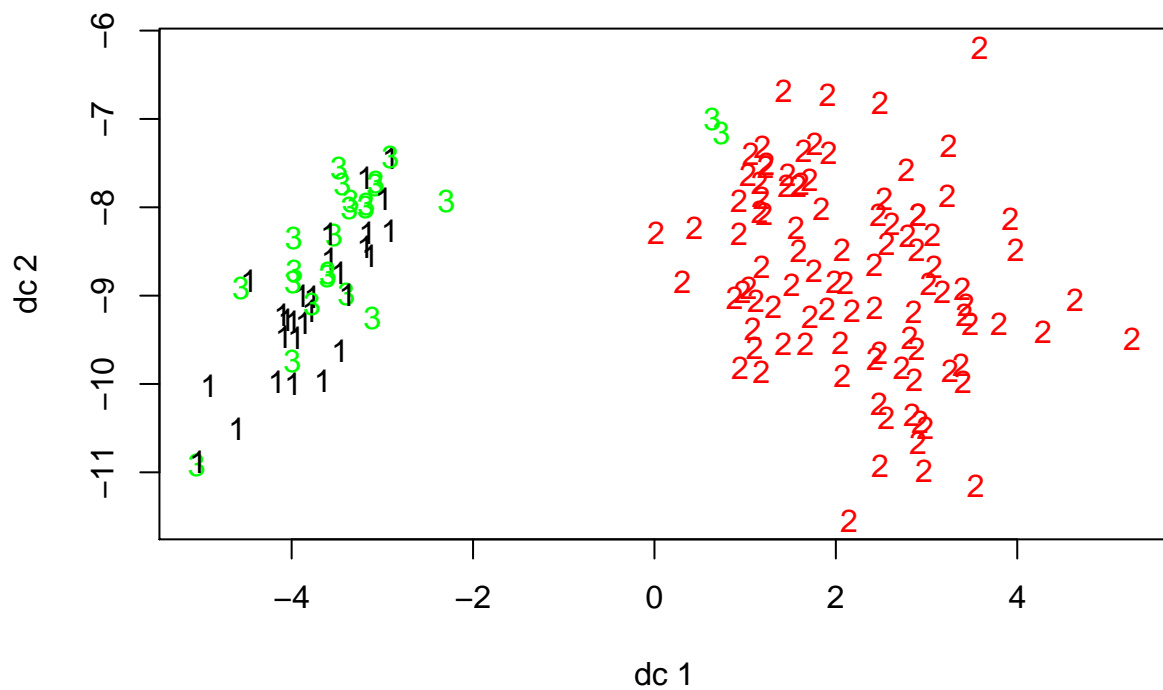
```

cluster2.scale <- kmeans_ft(scale(iris[,1:4]))
with(iris, pairs(scale(iris[,1:4]), col=c(1:3)[cluster2.scale$new_group]))

```



```
plotcluster(iris[-5], cluster2.scale$new_group)
```



```
set.seed(11)
k2 <- kmeans_ft(scale(iris[,1:4]))
k2
```

```
## $number_of_iter
## [1] 5
##
```

```
## $new_group
## [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
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 1 1 1 3 1 1 1 1 3 1 1 3 1 1 1 1
## [71] 3 1 1 1 3 3 1 1 1 1 1 1 1 1 3 3 1 1 3 1 1 3 1 1 3 1 1 1 3 3 3 1
## [106] 1 3 3 1 3 3 1 3 3 3 1 3 3 3 1 1 1 3 1 3 1 1 3 3 3 3 1 1 1 3 1 3 3
## [141] 3 3 1 3 1 3 1 1 3 1

d11 <- iris[,5][which(k2$new_group==1)]
accu11 <- length(d11[d11=="versicolor"])/length(d11)
d22 <- iris[,5][which(k2$new_group==2)]
accu22 <- length(d22[d22=="setosa"])/length(d22)
d33 <- iris[,5][which(k2$new_group==3)]
accu33 <- length(d33[d33=="virginica"])/length(d33)
accu11; accu22; accu33

## [1] 0.6363636
## [1] 1
## [1] 0.6666667
```

Even if we use the scaled iris data, the k-means function fails to cluster different species perfectly. The plot is similar to the unscaled iris data, the k-means function is only successful in distinguishing one type of species. In addition, we can check that when we use the scaled iris data, the accuracy is only improved in minor scale. In conclusion, while clustering data using k-means function is not majorly improved when using the scaled iris data versus original iris data.