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){
        for(j in 2:(n.col+1)){
            distance[i,1] <- sqrt(sum((new.data[i,j]-mat.mu[1,j])^2))
            distance[i,2] <- sqrt(sum((new.data[i,j]-mat.mu[2,j])^2))
            distance[i,3] <- sqrt(sum((new.data[i,j]-mat.mu[3,j])^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)
```

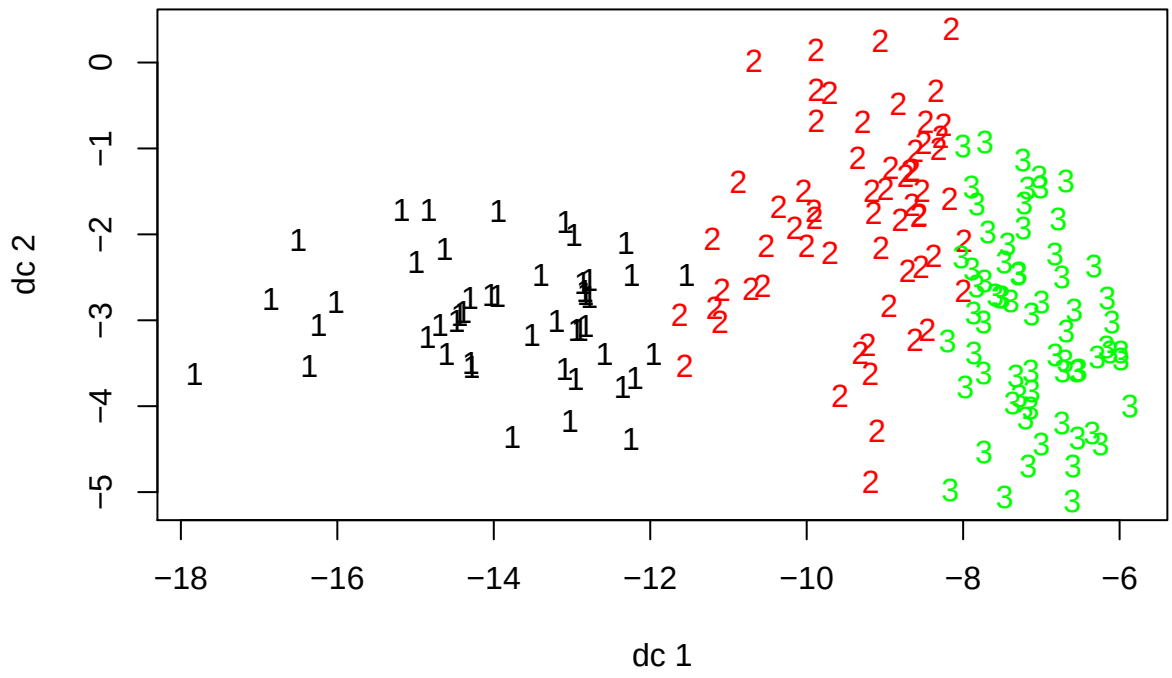
```
{
    mat.data <- data.matrix(data)
    n.row <- nrow(mat.data)
    n.col <- ncol(mat.data)

    # 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(10)
cluster <- kmeans_ft(wine[-1])
plotcluster(wine[-1], cluster$new_group)
```




```
cluster
```

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

```
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] 0.3225806
```

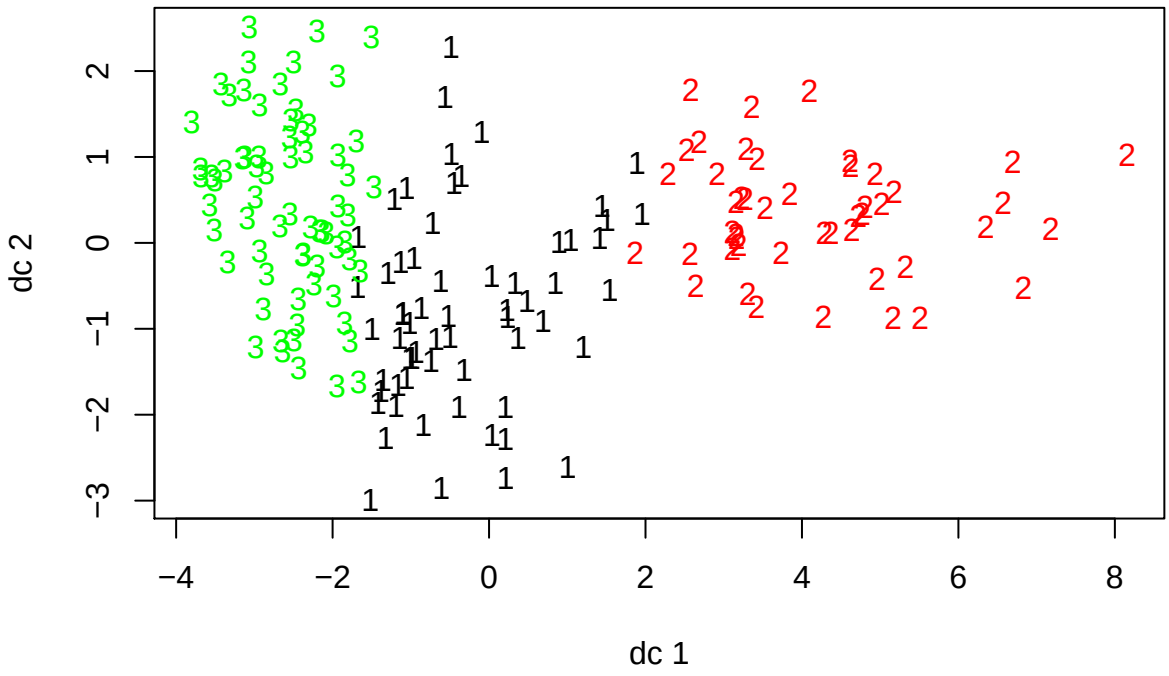
```
## [1] 0.2753623
```

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.

```
data.train <- scale(wine[-1])
set.seed(3)
cluster.scale <- kmeans_ft(data.train)
plotcluster(data.train, cluster.scale$new_group)
```




```
cluster.scale <- kmeans_ft(data.train)
type.g1.scale <- wine[,1][which(cluster.scale$new_group==1)]
accurate.scale.1 <- length(type.g1.scale[type.g1.scale==1])/length(type.g1.scale)
type.g2.scale <- wine[,1][which(cluster.scale$new_group==2)]
accurate.scale.2 <- length(type.g2.scale[type.g2.scale==2])/length(type.g2.scale)
type.g3.scale <- wine[,1][which(cluster.scale$new_group==3)]
```

```
accurate.scale.3 <- length(type.g3.scale[type.g3.scale==3])/length(type.g3.scale)
print(list(type.g1.scale, type.g2.scale, type.g3.scale))
```

```
## [[1]]
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [36] 1 1 1 1 1 1 1 1 1 1 1 1 2
## Levels: 1 2 3
##
## [[2]]
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3
## [36] 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## Levels: 1 2 3
##
## [[3]]
##  [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 2 2 2 2 2
## [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## Levels: 1 2 3
```

```
accurate.scale.1; accurate.scale.2; accurate.scale.3
```

```
## [1] 0.9787234
```

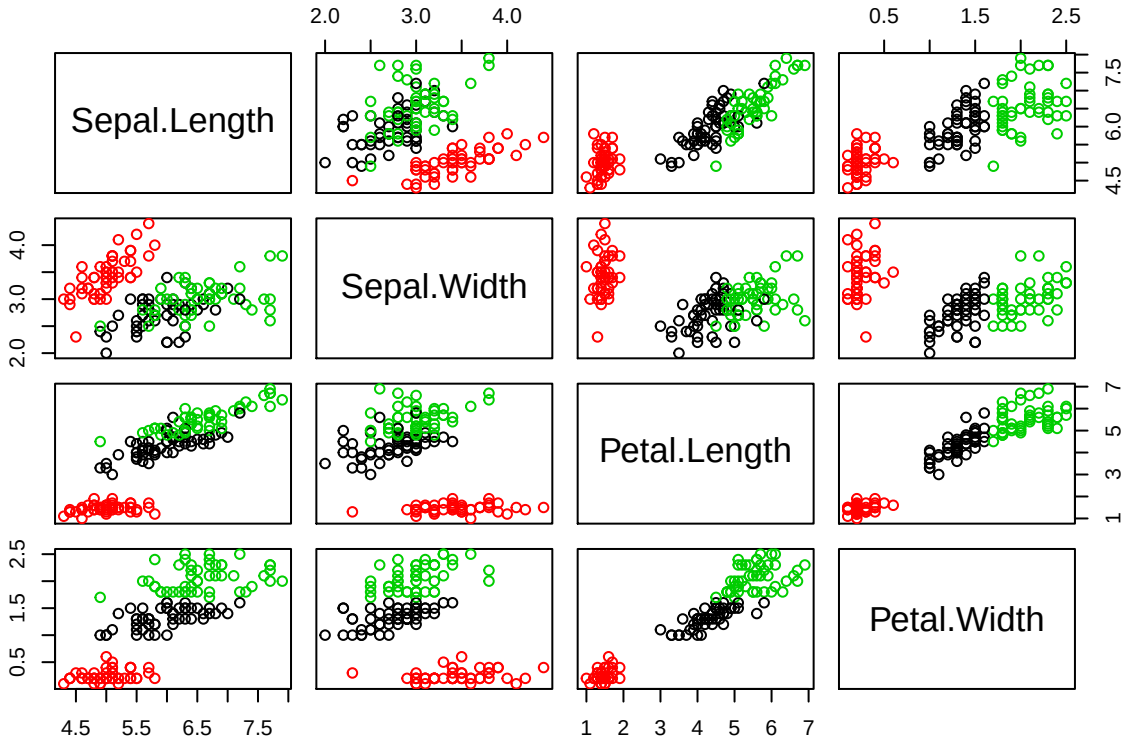
```
## [1] 0.3225806
```

```
## [1] 0.2753623
```

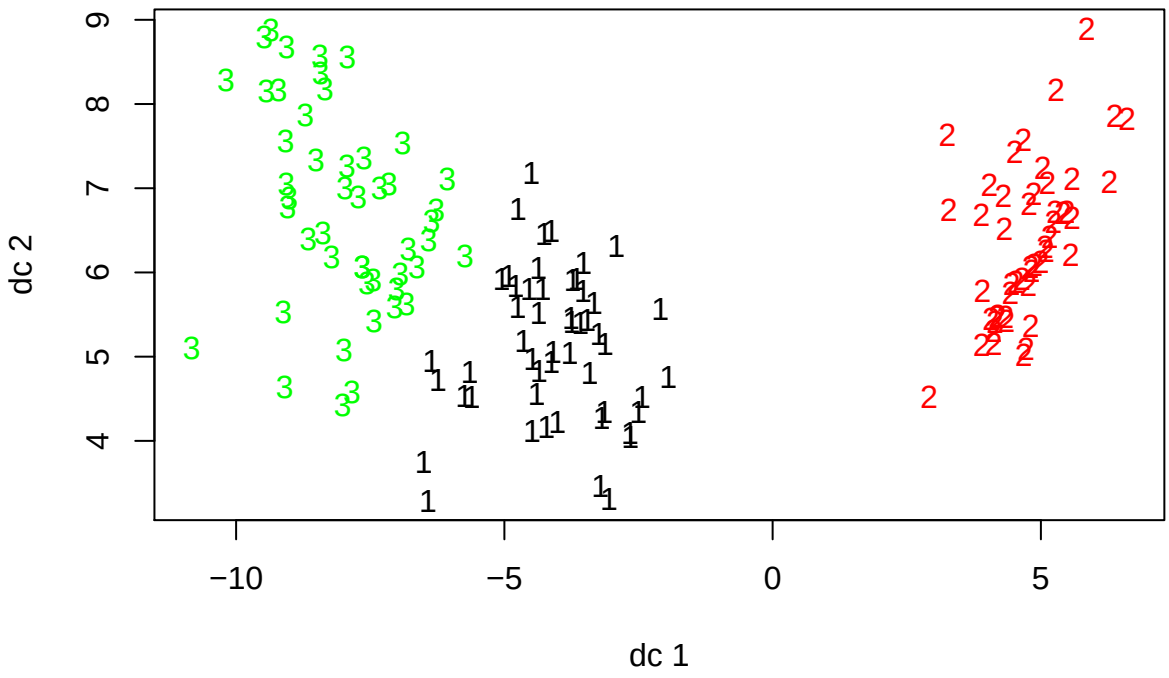
The scaled data improved the clustering results, but still failed to distinguish from Type 2 and wine Type 3 in the original wine data.

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

```
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] 0.9230769
```

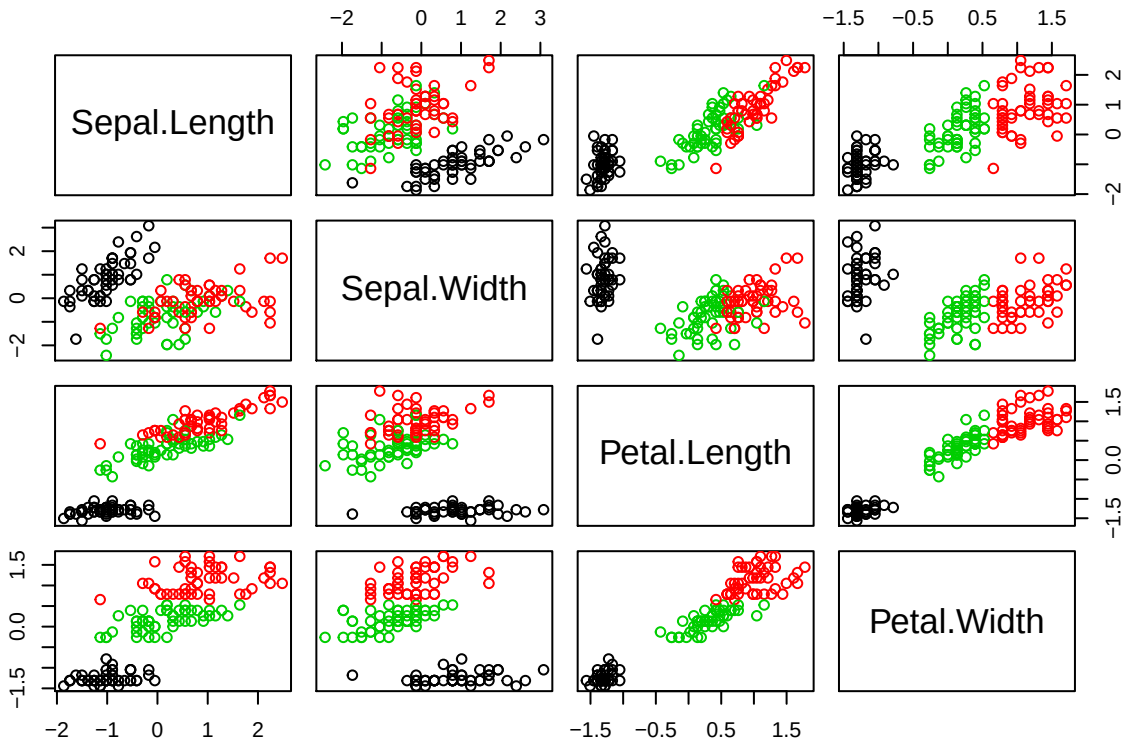
```
## [1] 1
```

```
## [1] 0.9583333
```

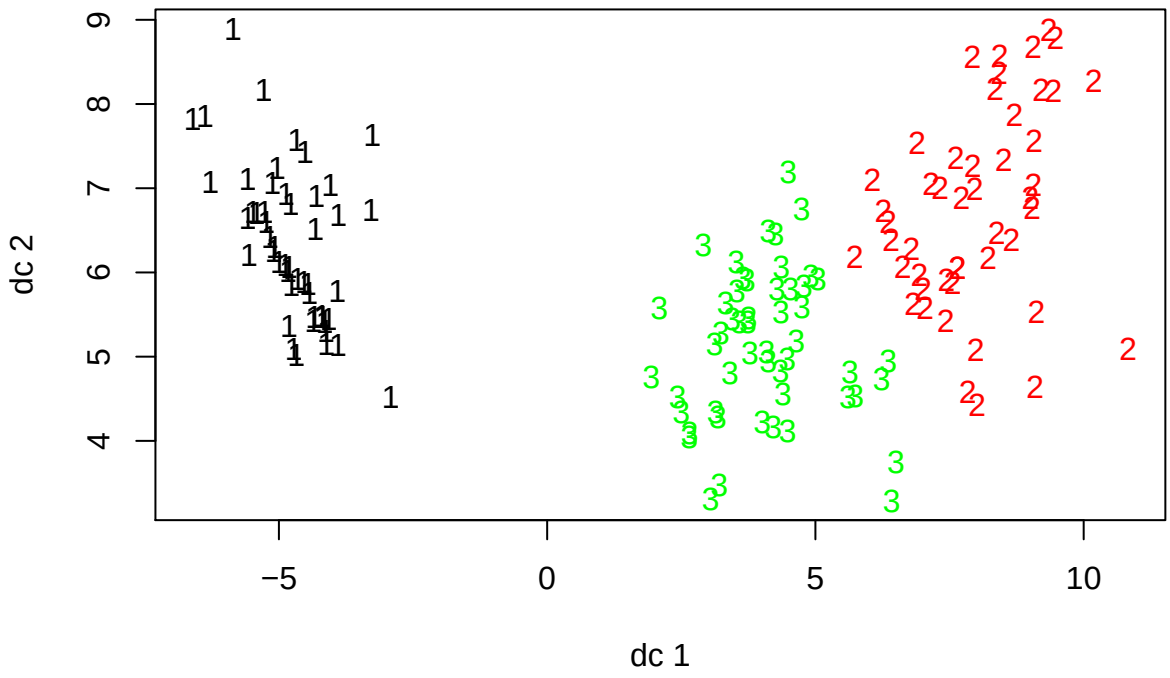
Now, let's look at the iris data set. As we can see in the plot, the data set is well seperated into three groups. In addition, if we calculate the accuracy rate of the result from k-means to the Species in the original iris data set, we can check that the accuracy rates for three groups are almost 1, indicating that our k-means function has perfectly seperated the data set into 3 groups.

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(10)
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 2 2 2 2
##  [36] 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
##  [71] 3 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3 3 3 3 3
## [106] 3 3 3 3 3 3 3 3 3 3 3 3 3 1 3 3 3 3 3 3 3 3 1 3 3 3 1 1 3 3 3 3 3
## [141] 3 3 3 3 3 3 3 3 3 3
```

```
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] 0.9230769
```

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

```
## [1] 0.9583333
```

Likewise, for the scaled data of iris, we can also check that k-means works well that it successfully grouped the data set into 3 clusters. If we calculate the accuracy rate of this result, we can check that the accuracy rates for each species are almost 1, as same as the previous result without scaling. That is, for this case, since the k-means in the original data set without scaling already almost perfectly classify the data, scaling does not have significant difference, but it still classifies the data almost perfectly.