K_Means

Boxin Zhao

10/30/2018

Load the data

```
data(wine, package="rattle")
```

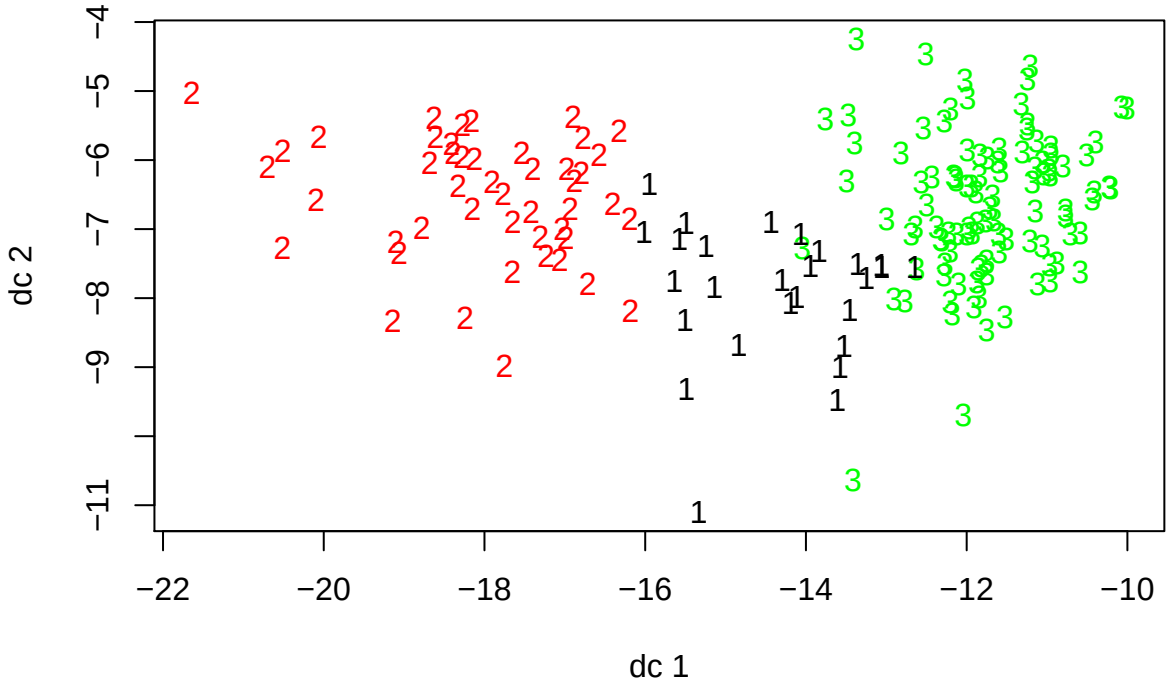
Define our k-means function

```
MyKMeans <- function(m, k, n.iteration=1000){
  # Self defined function to do K-Means
  #
  # Args:
  #   m: The data need to be clustered, should be matrix or data frame
  #   k: The number of clusters want to classify
  #   n.iteration: Number of iteration times
  #
  # Returns:
  #   A vector indicating the class of each observation
  m <- as.matrix(m)
  n <- dim(m)[1]  # number of observations
  class.record <- matrix(0, nrow=n.iteration, ncol=n)  # a matrix recording the
                                                       # class in each iteration
  center.record <- array(0, c(n.iteration, k, dim(m)[2]))  # an array recording the
                                                 # centers in each iteration

  # Randomly intialize the cluster centers.
  center.index <- sample(1:n, k, replace=FALSE)
  centers <- m[center.index, ]
  center.record[1, , ] <- centers

  for (t in 1:(n.iteration - 1)){
    center <- center.record[t, , ]

    # assignment step
    for (i in 1:n){
      observ <- as.vector(m[i, ])
      distance <- numeric(k)
      for (j in 1:k){
        center <- as.vector(centers[j, ])
        distance[j] <- sum((observ - center) ^ 2)
      }
      class.record[t, i] <- which.min(distance)
    }

    # update step
    for (j in 1:k){
      cluster.temp <- m[(class.record[t, ] == j), ]
      if (length(dim(cluster.temp)) > 1){
        center.record[t + 1, j, ] <- colMeans(cluster.temp)
      } else {
```

```
        center.record[t + 1, j, ] <- cluster.temp
      }
    }

  }

  t <- n.iteration
  center <- center.record[t, , ]

  for (i in 1:n){
      observ <- as.vector(m[i, ])
      distance <- numeric(k)
      for (j in 1:k){
        center <- as.vector(centers[j, ])
        distance[j] <- sum((observ - center) ^ 2)
      }
      class.record[t, i] <- which.min(distance)
    }

  return(class.record[t, ])
}
```

Visualize the clustering result

```
library('fpc')

k <- 3  # number of clusters

data.train <- wine[-1]
y.pred <- MyKMeans(data.train, k)
plotcluster(data.train, y.pred)
```



The clusters seem well seperated.

Before doing any analysis, we first align the predicted labels with the true labels. We do the alignment by following steps: For j = 1 to k, where k is the number of clusters, we choose the subsets of data which are predicted with label j. We then choose the first element of this subset, and check what is the true label of this element. If this true label has not been given to other subsets predicted with different j, we then assign the true label to this subset. If the label has been given to other subsets with different j, we check the second element, and again to assign the true label of second element to this subset if it is not used. If the second element does not work either, we keep doing this until we find an element can work. If none of the elements in this subset work, we just keep its orginal predicted label.

After the alignment, we calculate the prediction precision by just counting the precentage of right labelled observations.

We run KMeans 100 times and calculating the precision in each iteration, and report the mean of precision as the final result.

```
M <- 10
precision.record <- numeric(M)
data.train <- wine[-1]

for (t in 1:M){
  y.pred <- MyKMeans(data.train, k)
  y.pred.new <- y.pred
  y.true <- as.vector(wine[, 1])

  used.label <- c()
  for (j in 1:k){
```

```
    subsets.index <- which(y.pred == j)
    for (i in subsets.index){
      new.label <-y.true[i]
      if (is.element(new.label, used.label)){
      } else {
        used.label[j] <- new.label
        y.pred.new[subsets.index] <- new.label
        break
      }
    }
  }

  precison <- as.double(sum(y.pred.new == y.true)) / length(y.true)
  precision.record[t] <- precison
}

precison <- mean(precision.record)

print(paste("The predictiton precision is:", round(precison * 100, 2), "%"))
```

```
## [1] "The predictiton precision is: 40.67 %"
```

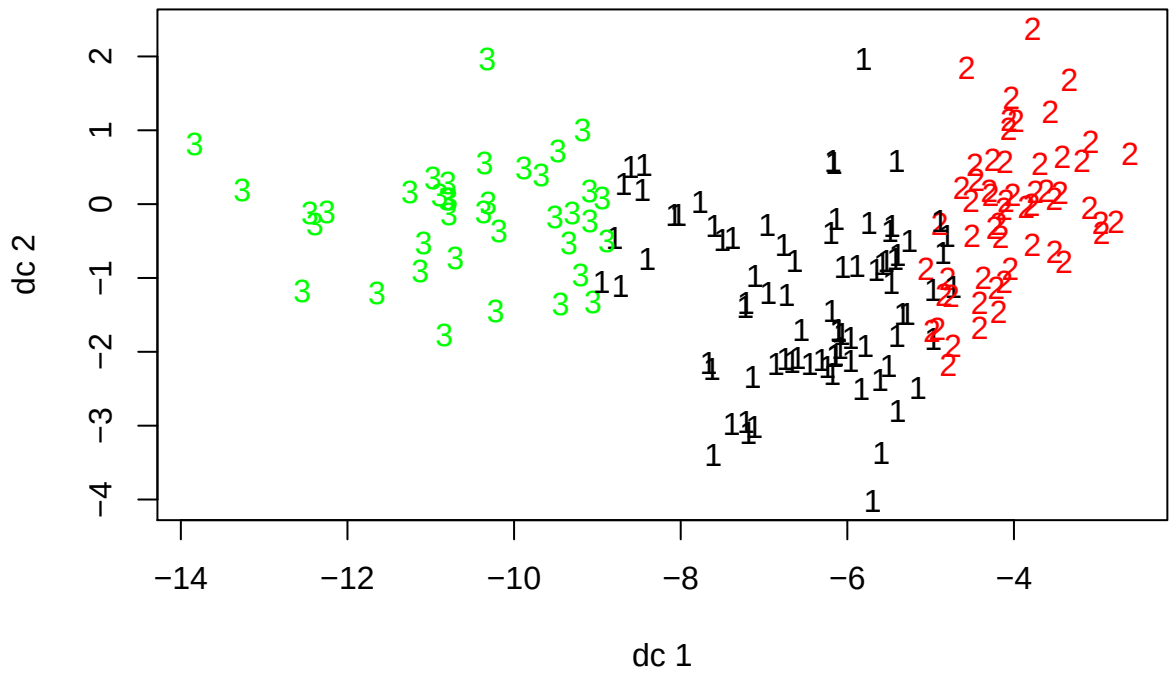
Then we see what how scale will change the data

```
data.train <- scale(wine[-1])
```

This function substract column mean from each column, and divide each column by its standard deviation, so that each column will have zero mean and unit standard deviation.

Visualize the clustering result

```
library('fpc')

k <- 3  # number of clusters

data.train <- wine[-1]
y.pred <- MyKMeans(data.train, k)
plotcluster(data.train, y.pred)
```



Calculate the prediction precision:

```
M <- 100
precision.record <- numeric(M)
data.train <- scale(wine[-1])

for (t in 1:M){
  y.pred <- MyKMeans(data.train, k)
  y.pred.new <- y.pred
  y.true <- as.vector(wine[, 1])

  used.label <- c()
  for (j in 1:k){
    subsets.index <- which(y.pred == j)
    for (i in subsets.index){
      new.label <-y.true[i]
      if (is.element(new.label, used.label)){
      } else {
        used.label[j] <- new.label
        y.pred.new[subsets.index] <- new.label
        break
      }
    }
  }

  precison <- as.double(sum(y.pred.new == y.true)) / length(y.true)
  precision.record[t] <- precison
```

```
}

precison <- mean(precision.record)

print(paste("The predictiton precision is:", round(precison * 100, 2), "%"))
```

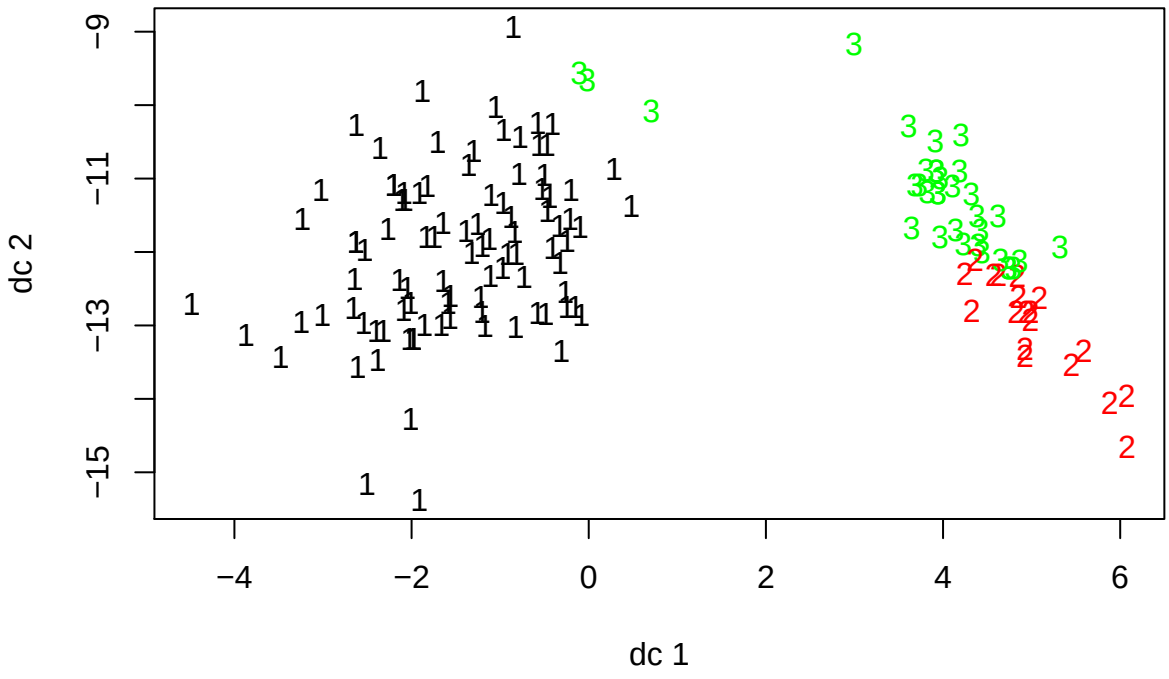
```
## [1] "The predictiton precision is: 39.43 %"
```

We see that the prediction precision decreases, so the scale procedure may actually hurt the clustering.

Then we use iris dataset to check our method. We first work without scaling

Visualize the prediction result

```
data(iris)

data.train <- iris[-5]
k <- length(unique(iris[, 5]))

y.pred <- MyKMeans(data.train, k)
plotcluster(data.train, y.pred)
```



Calculate the prediction precision:

```
M <- 100
precision.record <- numeric(M)
data.train <- iris[-5]

for (t in 1:M){
```

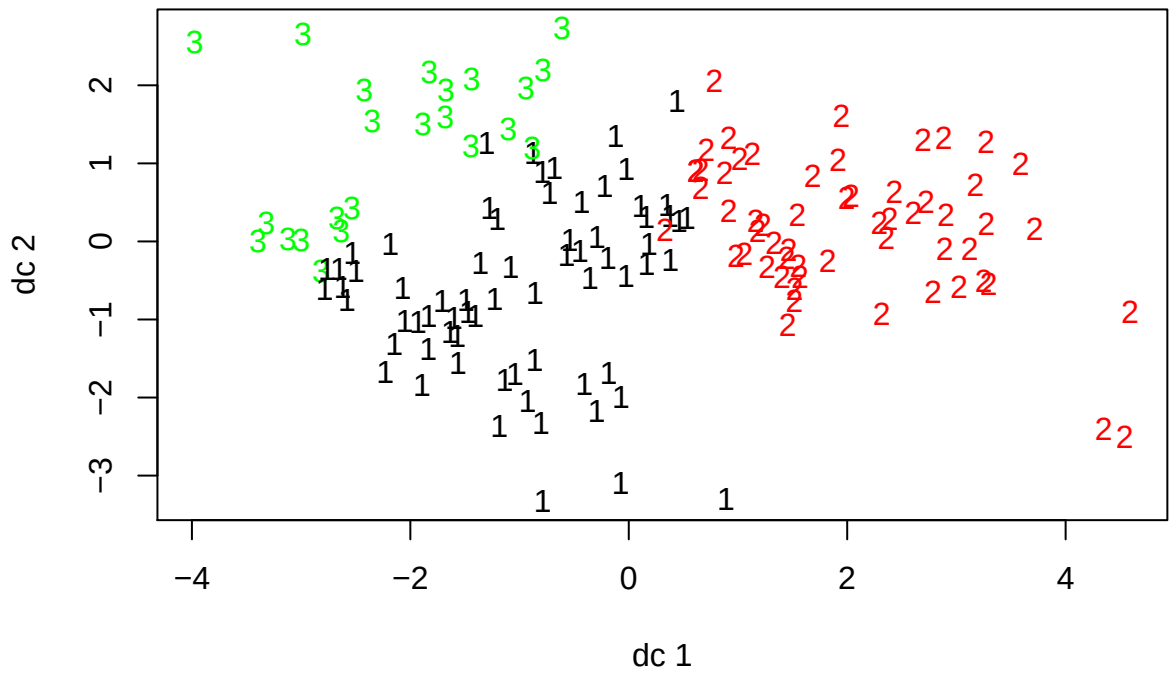
```
  y.pred <- MyKMeans(data.train, k)
  y.pred.new <- y.pred
  y.true <- as.vector(iris[, 5])

  used.label <- c()
  for (j in 1:k){
    subsets.index <- which(y.pred == j)
    for (i in subsets.index){
      new.label <-y.true[i]
      if (is.element(new.label, used.label)){
      } else {
        used.label[j] <- new.label
        y.pred.new[subsets.index] <- new.label
        break
      }
    }
  }

  precison <- as.double(sum(y.pred.new == y.true)) / length(y.true)
  precision.record[t] <- precison
}

precison <- mean(precision.record)

print(paste("The predictiton precision is:", round(precison * 100, 2), "%"))
```

```
## [1] "The predictiton precision is: 59.65 %"
```

We then see what scale will do

Visualization

```
data(iris)

data.train <- scale(iris[-5])
k <- length(unique(iris[, 5]))

y.pred <- MyKMeans(data.train, k)
plotcluster(data.train, y.pred)
```



Calculate the prediction precision:

```
M <- 100
precision.record <- numeric(M)
data.train <- scale(iris[-5])

for (t in 1:M){
  y.pred <- MyKMeans(data.train, k)
  y.pred.new <- y.pred
  y.true <- as.vector(iris[, 5])

  used.label <- c()
  for (j in 1:k){
    subsets.index <- which(y.pred == j)
    for (i in subsets.index){
      new.label <-y.true[i]
      if (is.element(new.label, used.label)){
      } else {
        used.label[j] <- new.label
        y.pred.new[subsets.index] <- new.label
        break
      }
    }
  }

  precison <- as.double(sum(y.pred.new == y.true)) / length(y.true)
  precision.record[t] <- precison
```

```
}

precison <- mean(precision.record)

print(paste("The predictiton precision is:", round(precison * 100, 2), "%"))
```

```
## [1] "The predictiton precision is: 50.43 %"
```

We see again that the scale procedure will decrease the prediction precision.