K-means clustering from scratch

Holly Capell

```r
euc.dist <- function(x1, x2){
  sqrt(sum((x1 - x2)** 2))
}


kmeans <- function(nReps, myScatterInput, myClusterNum, maxIter){
  # Create empty list to be filled with sums of the final euclidean distances from the centroids
  euclidSums <-  list()

  for (k in 1:nReps){
    # Create vector of cluster numbers
    clusterNums <- seq(1:myClusterNum)

    # Initialize cluster nums
    cluster_init <- sample(clusterNums, nrow(myScatterInput), replace = T)

    # Create cluster centroids by taking means of each group
    clusterCentroids <- bind_cols(myScatterInput, cluster = cluster_init) %>%
      group_by(cluster) %>%
      summarize_all(mean)

    cluster_new <- cluster_init

    count <- 0
    while (count < maxIter){
      cluster_old <- cluster_new

      # Create empty matrix to fill with distances
      dist_matrix <- matrix(0,nrow(myScatterInput), ncol=nrow((clusterCentroids)))

      # Turn cluster centroids into matris
      cluster_matrix <- as.matrix(clusterCentroids)

      # Iterate over each cluster centroid to calculate euclidean distances with myScatterInput
      for (i in 1:nrow(clusterCentroids)){
        differences <- t(myScatterInput) - cluster_matrix[i,-1]

        eucDist <- t(sqrt(colSums(differences^2)))
        dist_matrix[,i] <- eucDist

      }
    # Find the minimum cluster distance for each observation in order to reassign clusters
    cluster_new <- (sapply(seq(nrow(dist_matrix)), function(i) {
        which.min(dist_matrix[i,])
      }))

    # Recalculate cluster centroids
      clusterCentroids <- bind_cols(myScatterInput, cluster = cluster_new) %>%
        group_by(cluster) %>%
        summarize_all(mean)

      # Break once cluster centroid assignments stop changing
      if (sum(cluster_new - cluster_old) == 0){
        break
      }
      count = count + 1
    }

    # Create data frame of final cluster assignments
    cluster_DF <- data.frame(cluster_new)
    colnames(cluster_DF) <- c("cluster")

    # Match up final cluster assignments to cluster means
    cluster_mat <- as.matrix(left_join(cluster_DF, clusterCentroids, by="cluster"))
    cluster_mat <- cluster_mat[,-1]

    # Compute the difference between observations and final centroid mean
    differences <- t(myScatterInput) - t(cluster_mat)

    # Compute eculidean distances and sum them
    eucDist <- t(sqrt(colSums(differences^2)))
    eucSum <- sum(eucDist)

    euclidSums[[k]] <- eucSum

  }
  # Find the minimum eucliean sum
  minSum <- min(unlist(euclidSums))

  if (ncol(myScatterInput) == 2){
    myScatterInput$cluster <- cluster_new
    colnames(myScatterInput) <- c("x1", "x2", "cluster")
    plot <- ggplot(myScatterInput, aes(x =x1, y=x2))+
      geom_point(aes(color=factor(cluster)))

    return(plot)

  } else if (ncol(myScatterInput) == 3){
    myScatterInput$cluster <- cluster_new
    colnames(myScatterInput) <- c("x1", "x2", "x3", "cluster")
    colors <- c("red", "blue", "green", "orange", "yellow", "purple", "pink",
                "coral", "yellow", "navyblue", "turquoise", "olivedrab2")

    colors <- colors[as.numeric(myScatterInput$cluster)]
    plot <- scatterplot3d(myScatterInput[,1:3], color = colors)
    return(plot)
  }
  return (paste("Min Euclidean Sum: ",minSum))
}
```

```
# TEST DATA 1
set.seed(101)
```

```
myScatterInput <- data_frame(myCol_01 = runif(100000, -1, 1))
myClusterNum <- 2

kmeans(nReps = 5, myScatterInput = myScatterInput, myClusterNum = myClusterNum, maxIter = 10000)
```

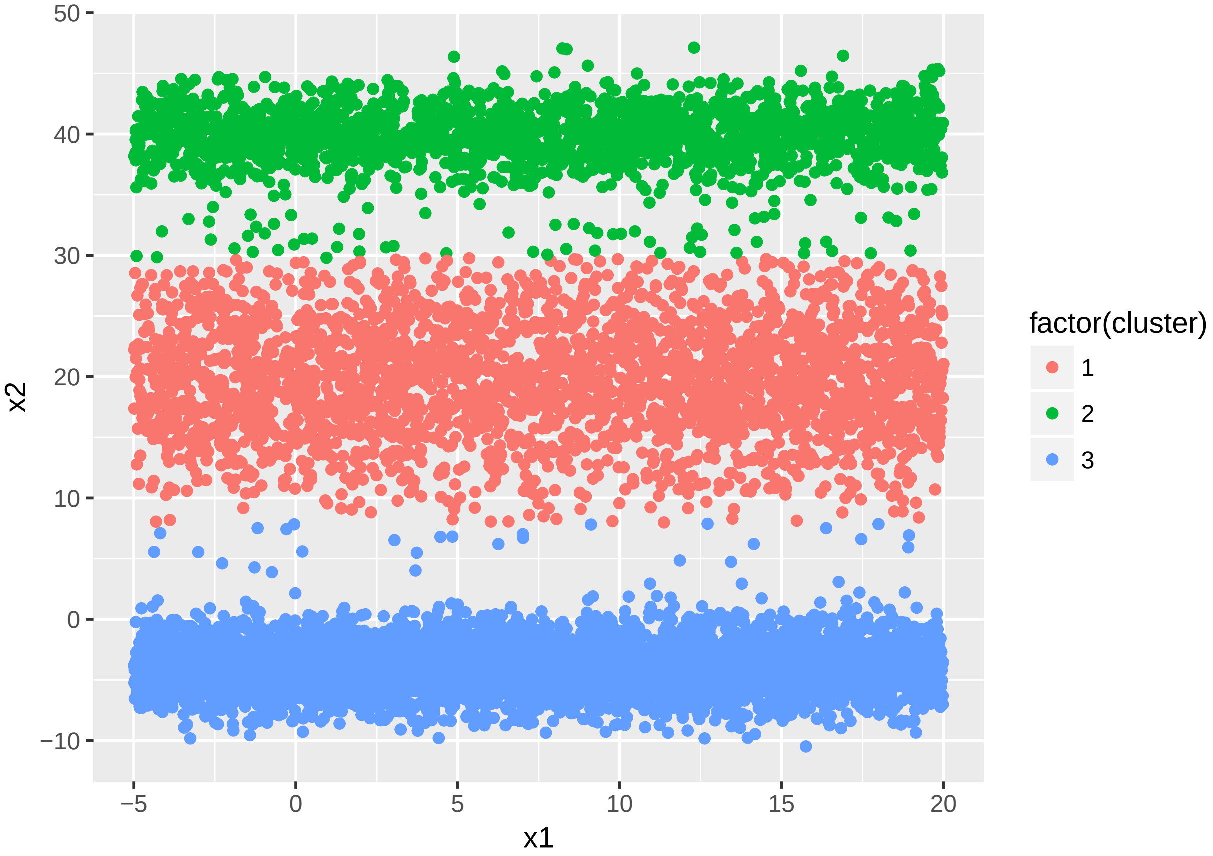
```
## [1] "Min Euclidean Sum:  24862.2309460583"
```

```
microbenchmark(kmeans(1, myScatterInput, myClusterNum, 10000),times = 2)
```

```
## Unit: seconds
##                                             expr      min       lq    mean
##  kmeans(1, myScatterInput, myClusterNum, 10000) 2.471587 2.471587 2.68203
##    median       uq      max neval
##  2.68203 2.892474 2.892474     2
```

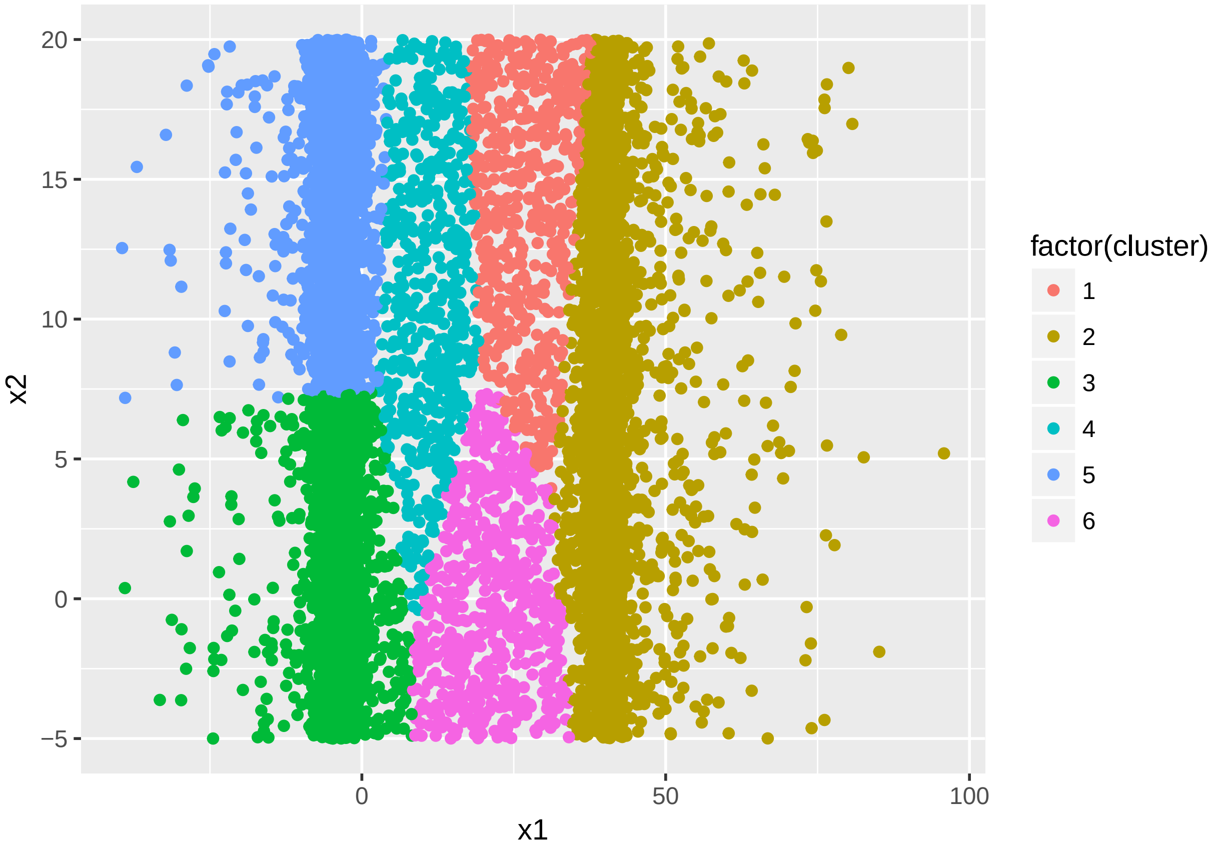
```
# TEST DATA 1
set.seed(103)
myScatterInput <- data_frame(myCol_01 = runif(10000, -5, 20), myCol_02 = c(rnorm(3000, 20, 5), rnorm(500
myClusterNum <- 3

kmeans(nReps = 5, myScatterInput = myScatterInput, myClusterNum = myClusterNum, maxIter = 10000)
```



```
# TEST DATA 2
set.seed(104)
myScatterInput <- data_frame(myCol_01 = c(rnorm(3000, 20, 20), rnorm(5000, -4, 2), rnorm(2000, 40, 2)),
myClusterNum <- 6

kmeans(nReps = 5, myScatterInput = myScatterInput, myClusterNum = myClusterNum, maxIter = 10000)
```


20
15
10
5
0
−5
x2
0
50
100
x1
factor(cluster)
● 1
● 2
● 3
● 4
● 5
● 6