Kmeans

Louise Lai

August 14, 2018

Function

```r
myKMeans <- function(myScatterInputOriginal, myClusterNum, numIter){
  totalDistances <- Inf

  for(i in 1:numIter){
    myScatterInput <- as.matrix(myScatterInputOriginal)
    n <- nrow(myScatterInput) # n = rows
    m <- ncol(myScatterInput) # m = dimensions

    # 1) random assignment
    myScatterInput <- cbind(myScatterInput, clusterAssignment=rep_len(1:myClusterNum, n))

    swapped = T
    while(swapped == T){
      # 2) compute centroids
      centroids <- matrix(data=NA, nrow=myClusterNum, ncol=m)
      if(m > 1){ # this is due to some funky subsetting thing
        centroids <- as.matrix(aggregate(list(myScatterInput), by=list(myScatterInput[,'clusterAssignme
      } else {
        centroids <- as.matrix(aggregate(list(myScatterInput), by=list(myScatterInput[,'clusterAssignme
      }

       #in case empty clusters arise, we need to update number of centroids
      myClusterNum <- nrow(centroids)

      # 3) distance from each data point to centriod
      # we use the wonderful cdist package, which calculates difference between two matrices
      newClusters <- vector(mode="double", length=n)
      for(i in 1:n){
        distPointCentroids <- rdist::cdist(X = centroids[1:myClusterNum,,drop=F], Y=myScatterInput[i, 1
        newClusters[i] <- which.min(distPointCentroids)
        # the target centriod is the one with min dist
      }

      # 4) assign to centroid
      # if previous clusters don't equal new clusters, swap and indicate swap occured
      swapped = F
      if(!identical(myScatterInput[,'clusterAssignment'], newClusters)){
        myScatterInput[,'clusterAssignment'] = newClusters
        swapped=T
      }
    }

    # 5) get sum of distances
    distances <- vector(mode="double", length=myClusterNum)
    for(i in 1:myClusterNum){
      clusterOfPoints <- as.data.frame(myScatterInput) %>% filter(clusterAssignment == i) %>% select(-cl
      distances[i] <- sum(rdist::cdist(X = centroids[1:myClusterNum,,drop=F], Y=as.matrix(clusterOfPoin
    }
    newTotalDistances <- sum(distances)
  }

  # 6) assign lowest distance
  if(newTotalDistances < totalDistances){
    totalDistances <- newTotalDistances
  }

  # 7) if 2D, plot
  if(m==2){
    print(ggplot(as.data.frame(myScatterInput)) +
            geom_point(aes(x=myCol_01, y=myCol_02, color=clusterAssignment)) +
            geom_point(data=as.data.frame(centroids), aes(x=myCol_01, y=myCol_02), size=6, pch=13) +
            scale_color_continuous(breaks = c(1:myClusterNum)) +
            theme_classic())
  }

  # finally, print the sum of distances
  print(totalDistances)
}
```

```
# TEST DATA 1
set.seed(101)
myScatterInput <- data_frame(myCol_01 = runif(100000, -1, 1))
myClusterNum <- 2

# timing
microbenchmark(myFunc=myKMeans(myScatterInput, myClusterNum, 2), times=1)
```

```
## [1] 124871.1
```

```
## Unit: seconds
##    expr      min       lq     mean   median       uq      max neval
##  myFunc 37.93229 37.93229 37.93229 37.93229 37.93229 37.93229     1
```