MSAN 593 HW4

Fei Liu

August 09, 2017

Problem 1

```r
first_replace <- c('stinky','lumpy','buttercup','gidget','crusty','greasy',
                   'fluffy','cheeseball', 'chim-chim','poopsie','flunky','booger',
                   'pinky','zippy','goober','doofus','slimy','loopy','snotty','falafel',
                   'dorkey','squeezit','oprah','skipper','dinky','zsa-zsa')
second_replace <- c('diaper','toilet','giggle','bubble','girdle','barf','lizard','waffle',
                    'cootie','monkey','potty','liver','banana','rhino','burger','hamster',
                    'toad','gizzard','pizza','gerbil', 'chicken','pickle','chuckle','tofu',
                    'gorilla','stinker')
third_replace <- c('head','mouth','face','nose','tush','breath','pants','shorts','lips',
                   'honker','butt','brain','tushie','chunks','hiney','biscuits','toes',
                   'buns','fanny','sniffer','sprinkles','kisser', 'squirt','humperdinck',
                   'brains','juice')

myfun <- function(name){
  #if name contains non-chracter return error
  if (regexpr('[^A-Z a-z]', name) > 0) return("Error: name is not conforming.")

  y <- strsplit(name, "\\s+")[[1]]
  #if name length > 2 words return error
  if(length(y) != 2) return("Error: name is not conforming.")

  first_old <- toupper(substr(y[1], start = 1, stop = 1))
  last_old_s <- toupper(substr(y[2], start = 1, stop = 1))
  last_old_e <- toupper(substr(y[2], start = nchar(y[2]) , stop = nchar(y[2])))

  #Capitalize only the first character
  proper=function(x) paste0(toupper(substr(x, 1, 1)), tolower(substring(x, 2)))

  #Concatenate new first name and last name
  first_new <- proper(first_replace[which(LETTERS == first_old)])
  last_half.1 <- proper(second_replace[which(LETTERS == last_old_s)])
  last_half.2 <- tolower(third_replace[which(LETTERS == last_old_e)])
  new_name <-  paste(first_new, " ", last_half.1, last_half.2, sep="")
  return(new_name)
}
test_input <- c("Fei Liu", "Paul Intrevado", "David Uminksy", "Terence Parr", "Jeff Hamrick",
                "paul intrevado", "Intrevado, Paul", "Intrevad0 Paul", "Queen Elizabeth II",
                "Queen Elizabeth 2nd","Queen Elizabeth 2", "John Paul Euclid Rumpel",
                "britishDudeThatSitsInTheBackOfTheClass")
test_output <- sapply(test_input, myfun, USE.NAMES = F)
test_output
```

```
##  [1] "Greasy Liversprinkles"          "Doofus Cootiehiney"
##  [3] "Gidget Chickenbrains"           "Falafel Hamsterbuns"
##  [5] "Poopsie Wafflebutt"             "Doofus Cootiehiney"
##  [7] "Error: name is not conforming." "Error: name is not conforming."
##  [9] "Error: name is not conforming." "Error: name is not conforming."
## [11] "Error: name is not conforming." "Error: name is not conforming."
## [13] "Error: name is not conforming."
```

Problem 2

Loading packages

```r
library(microbenchmark)
library(plotly)
library(tidyverse)
library(magrittr)
library(rdist)
library(scatterplot3d)
```

My Kmeans function

```r
Mykmeans <- function(myScatterInput, myClusterNum, nReps, maxIter) {

  minS <- Inf
  datavar <- names(myScatterInput)
  n <- nrow(myScatterInput)

  for (i in 1:nReps){

    myScatterInput['cluster']<- sample(1:myClusterNum, n, replace = T)
    iter <- 0
    while (iter <= maxIter){
      cluster.prev <- myScatterInput['cluster']
      centriod <- myScatterInput %>% group_by(cluster) %>% summarise_all(mean)
      cdist_try <- cdist(myScatterInput[,datavar],centriod[,-1])
      myScatterInput['cluster'] <- apply(cdist_try, 1, which.min)

      if (identical(cluster.prev, myScatterInput['cluster']) == T | iter == maxIter) {
        Eucli.dist <- sum(apply(cdist_try, 1, min))
        break}
      iter <- iter + 1
    }
    if (Eucli.dist < minS) {
      minS <- Eucli.dist
      curr_out <- myScatterInput
    }
  }
  #generate 2d plot
  # if (ncol(curr_out) == 3) {
  #    curr_out['cluster'] <- as.factor(as.character(curr_out$cluster))
  #    p <- ggplot(curr_out, aes(x = curr_out[,1], y = curr_out[,2], color = cluster)) + geom_point(siz
  #    print (p)
  #     }
  # #generate 3d plot
  # if (ncol(curr_out) == 4) {
  #    curr_out['cluster'] <- as.factor(as.character(curr_out$cluster))
  #    #plot_ly (myout, type = 'scatter3d' , x = ~V1 , y = ~V2 , z = ~V3,color = ~factor, mode = 'marke
  #    colors <- c("#999999", "#E69F00", "#56B4E9")
  #    colors <- colors[as.numeric(curr_out$cluster)]
  #    p <- scatterplot3d(curr_out[,1:3], pch = 16, color=colors)
  #    print(p)
  #    }
  return (minS) }
```

Test data

```r
# TEST DATA 1
set.seed(101)
myScatterInput1 <- data_frame(myCol_01 = runif(100000, -1, 1))
myClusterNum1 <- 2
# TEST DATA 2
set.seed(102)
myScatterInput2 <- data_frame(myCol_01 = runif(100000, -1, 1))
myClusterNum2 <- 4
# TEST DATA 3
set.seed(103)
myScatterInput3 <- data_frame(myCol_01 = runif(10000, -5, 20), myCol_02 = c(rnorm(3000, 20, 5), rnorm(5
myClusterNum3 <- 3
# TEST DATA 4
set.seed(104)
myScatterInput4 <- data_frame(myCol_01 = c(rnorm(3000, 20, 20), rnorm(5000, -4, 2), rnorm(2000, 40, 2))
myClusterNum4 <- 6
# TEST DATA 5
set.seed(105)
myScatterInput5 <- data_frame(myCol_01 = c(rnorm(3000, 20, 20), rnorm(5000, -4, 2), rnorm(2000, 40, 2))
                              myCol_02 = runif(10000, -5, 20),
                              myCol_03 = runif(10000, -100, 100),
                              myCol_04 = c(runif(4000, -5, 20), rnorm(6000)),
                              myCol_05 = runif(10000, -10, 200),
                              myCol_06 = rnorm(10000, -300, 1000),
                              myCol_07 = rnorm(10000, -1000000, 1000000),
                              myCol_08 = rnorm(10000, 30, 2))
myClusterNum5 <- 3
# TEST DATA 6
set.seed(106)
myScatterInput6 <- data_frame(myCol_01 = c(rnorm(3000, 20, 20), rnorm(5000, -4, 2), rnorm(2000, 40, 2))
                              myCol_02 = runif(10000, -5, 20),
                              myCol_03 = runif(10000, -100, 100),
                              myCol_04 = c(runif(4000, -5, 20), rnorm(6000)),
                              myCol_05 = runif(10000, -10, 200),
                              myCol_06 = rnorm(10000, -300, 1000),
                              myCol_07 = rnorm(10000, -1000000, 1000000),
                              myCol_08 = rnorm(10000, 30, 2))
myClusterNum6 <- 12
```

```
microbenchmark(Mykmeans(myScatterInput1, myClusterNum1, 10, 10000), times = 10)
```

```
## Unit: seconds
##                                                 expr      min       lq
##  Mykmeans(myScatterInput1, myClusterNum1, 10, 10000) 25.47442 25.99521
##      mean   median       uq      max neval
##  26.41247 26.09404 26.80561 27.74754    10
```

```
microbenchmark(Mykmeans(myScatterInput2, myClusterNum2, 10, 10000), times = 10)
```

```
## Unit: seconds
##                                                 expr      min     lq
##  Mykmeans(myScatterInput2, myClusterNum2, 10, 10000) 94.86307 95.453
##      mean   median       uq      max neval
##  96.21242 95.70065 97.39141 98.10455    10
```

```
microbenchmark(Mykmeans(myScatterInput3, myClusterNum3, 10, 10000), times = 10)
```

```
## Unit: seconds
##                                                 expr     min     lq
##  Mykmeans(myScatterInput3, myClusterNum3, 10, 10000) 4.91622 5.0345
##      mean   median       uq      max neval
##  5.132553 5.097449 5.208731 5.434948    10
```

```
microbenchmark(Mykmeans(myScatterInput4, myClusterNum4, 10, 10000), times = 10)
```

```
## Unit: seconds
##                                                 expr      min       lq
##  Mykmeans(myScatterInput4, myClusterNum4, 10, 10000) 13.09681 13.29308
##      mean   median       uq      max neval
##  15.63153 14.70142 15.55055 25.58258    10
```

```
microbenchmark(Mykmeans(myScatterInput5, myClusterNum5, 10, 10000), times = 10)
```

```
## Unit: seconds
##                                                 expr    min       lq
##  Mykmeans(myScatterInput5, myClusterNum5, 10, 10000) 5.0615 5.241671
##      mean   median       uq      max neval
##  5.432606 5.323272 5.394392 6.304569    10
```

```
microbenchmark(Mykmeans(myScatterInput6, myClusterNum6, 10, 10000), times = 10)
```

```
## Unit: seconds
##                                                 expr      min       lq
##  Mykmeans(myScatterInput6, myClusterNum6, 10, 10000) 55.77849 56.33869
##      mean   median       uq      max neval
##  56.66083 56.50269 56.79629 58.17123    10
```

Kmeans function with plot

```
Mykmeans <- function(myScatterInput, myClusterNum, nReps, maxIter) {

  minS <- Inf
  datavar <- names(myScatterInput)
  n <- nrow(myScatterInput)
```

```r
  for (i in 1:nReps){

    myScatterInput['cluster']<- sample(1:myClusterNum, n, replace = T)
    iter <- 0
    while (iter <= maxIter){
      cluster.prev <- myScatterInput['cluster']
      centriod <- myScatterInput %>% group_by(cluster) %>% summarise_all(mean)
      cdist_try <- cdist(myScatterInput[,datavar],centriod[,-1])
      myScatterInput['cluster'] <- apply(cdist_try, 1, which.min)

      if (identical(cluster.prev, myScatterInput['cluster']) == T | iter == maxIter) {
        Eucli.dist <- sum(apply(cdist_try, 1, min))
        break}
      iter <- iter + 1
    }
    if (Eucli.dist < minS) {
      minS <- Eucli.dist
      curr_out <- myScatterInput
    }
  }
  #generate 2d plot
  if (ncol(curr_out) == 3) {
     curr_out['cluster'] <- as.factor(as.character(curr_out$cluster))
     p <- ggplot(curr_out, aes(x = curr_out[,1], y = curr_out[,2], color = cluster)) + geom_point(size=
     print (p)
      }
  #generate 3d plot
  if (ncol(curr_out) == 4) {
     curr_out['cluster'] <- as.factor(as.character(curr_out$cluster))
     #plot_ly (myout, type = 'scatter3d' , x = ~V1 , y = ~V2 , z = ~V3,color = ~factor, mode = 'markers
     colors <- c("#999999", "#E69F00", "#56B4E9")
     colors <- colors[as.numeric(curr_out$cluster)]
     p <- scatterplot3d(curr_out[,1:3], pch = 16, color=colors)
     print(p)
     }
  return (minS) }
```

```r
Mykmeans(myScatterInput1, myClusterNum1, 10, 10000)
```

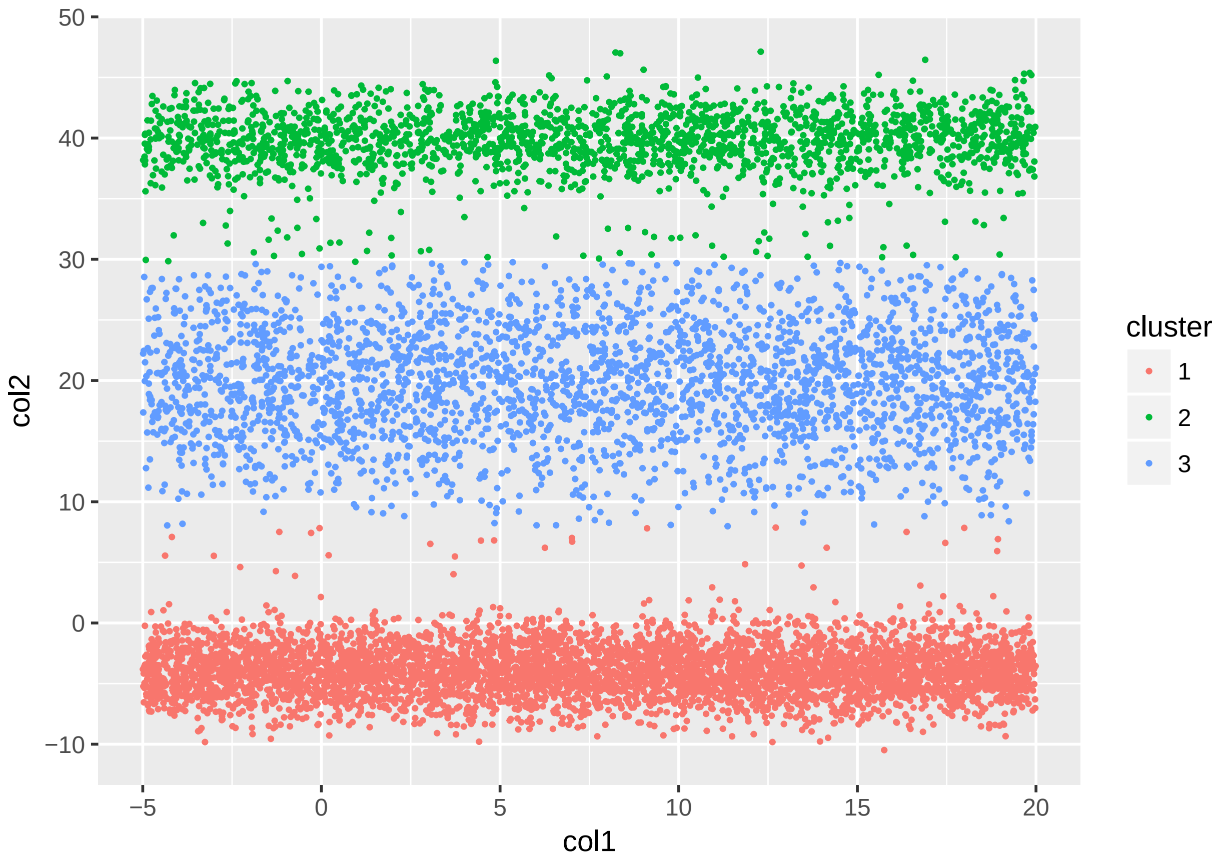
```
## [1] 24862.23
```

```r
Mykmeans(myScatterInput2, myClusterNum2, 10, 10000)
```

```
## [1] 12518.26
```

```r
Mykmeans(myScatterInput3, myClusterNum3, 10, 10000)
```

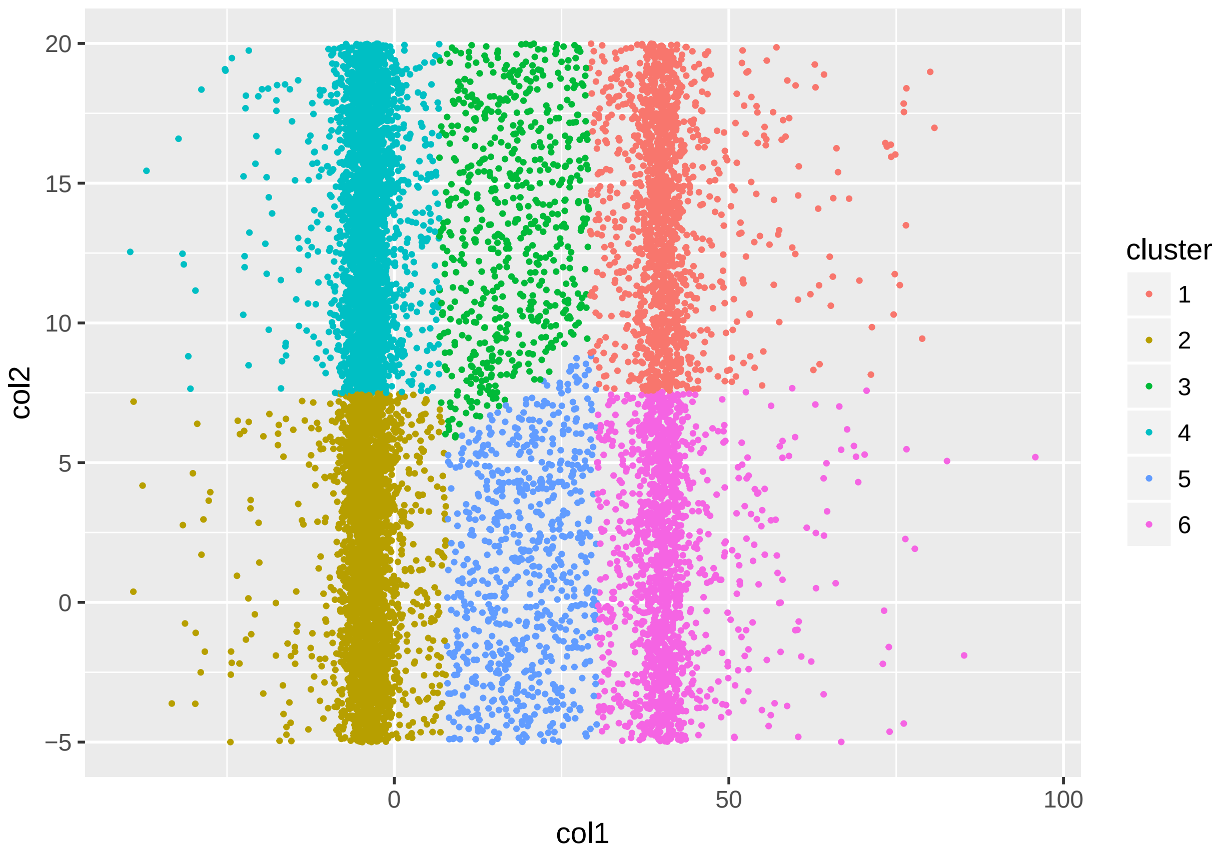
```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to c
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to c
```



```
## [1] 70111.94
```

```
Mykmeans(myScatterInput4, myClusterNum4, 10, 10000)
```

```
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
## Don't know how to automatically pick scale for object of type tbl_df/tbl/data.frame. Defaulting to co
```



```
## [1] 50138.36
```

```
Mykmeans(myScatterInput5, myClusterNum5, 10, 10000)
```

```
## [1] 3437723848
```

```
Mykmeans(myScatterInput6, myClusterNum6, 10, 10000)
```

```
## [1] 1006463434
```