clustering_ni.R

NRI

Mon Jun 27 21:52:39 2016

```
setwd("~/Desktop/practicum/venga_practicum/")
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(ggplot2)
library(data.table)
```

```
##
## Attaching package: 'data.table'

## The following objects are masked from 'package:dplyr':
##
##     between, last
```

```
library(tidyr)
library(reshape2)
```

```
##
## Attaching package: 'reshape2'

## The following objects are masked from 'package:data.table':
##
##     dcast, melt
```

```
library(stats)
library(caret)
```

```
## Loading required package: lattice
```

```r
library(corrplot)
library(fpc)
library(cluster)
options( java.parameters = "-Xmx4g" )

user <- fread("user_view_3865_ni.csv",check.names = T,
              header = T)
#keep these for later reference merging
user.rownames <- data.frame(rows = row.names(user),loyalty_user_id = user$loyalty_user_id)

#separate sets for repeat vs non repeat. repeat defined as anyone over 1 visit

#repeat
repeat.user.ind <- with(user,num_of_repeats == 1)
repeat.user <- user[!repeat.user.ind,]

#first time
firsttime.user.ind <- with(user,num_of_repeats > 1)
firsttime.user <- user[!firsttime.user.ind,]

#get user data ready for clustering, first removing variables that cannot be made numeric

#sapply does not work with data.table so have to turn it into a data.frame
users.cluster <- as.data.frame(user)
user.numeric <- users.cluster[sapply(users.cluster,is.numeric)]
str(user.numeric)
```

```
## 'data.frame':    50016 obs. of  73 variables:
##  $ num_of_repeats            : int  1 1 1 1 1 1 1 1 4 1 ...
##  $ min_party_size            : int  2 8 6 2 6 4 4 4 2 3 ...
##  $ max_party_size            : int  2 8 6 2 6 4 4 4 2 3 ...
##  $ avg_party_size            : num  2 8 6 2 6 4 4 4 2 3 ...
##  $ min_cover_count           : int  2 4 6 2 6 1 4 4 1 3 ...
##  $ max_cover_count           : int  2 4 6 2 6 1 4 4 2 3 ...
##  $ avg_cover_count           : num  2 4 6 2 6 1 4 4 1.5 3 ...
##  $ min_discount              : int  0 0 -28 0 0 -27 0 0 0 0 ...
##  $ max_discount              : int  0 0 -28 0 0 -27 0 0 0 0 ...
##  $ avg_discount              : num  0 0 -28 0 0 -27 0 0 0 0 ...
##  $ min_spend                 : num  115 436 375 100 188 ...
##  $ max_spend                 : num  115 436 375 100 188 ...
##  $ avg_spend                 : num  115 436 375 100 188 ...
##  $ min_diffdays_madeon_open  : num  2.035 3.239 40.335 0.823 7.861 ...
##  $ max_diffdays_madeon_open  : num  2.035 3.239 40.335 0.823 7.861 ...
##  $ avg_diffdays_madeon_open  : num  2.035 3.239 40.335 0.823 7.861 ...
##  $ min_diffmins_rsvp_open    : num  7.98 17.18 2.48 29.77 10.77 ...
##  $ max_diffmins_rsvp_open    : num  7.98 17.18 2.48 29.77 10.77 ...
##  $ avg_diffmins_rsvp_open    : num  7.98 17.18 2.48 29.77 10.77 ...
##  $ avg_weekday_visits        : num  0 1 0 1 1 1 0 1 0.5 0 ...
##  $ avg_weekend_visits        : num  1 0 1 0 0 0 1 0 0.5 1 ...
##  $ avg_breakfast_visits      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ avg_lunch_visits          : num  0 0 0 0 1 0 0 1 0.5 0 ...
##  $ avg_dinner_visits         : num  1 1 1 1 0 1 1 0 0.5 1 ...
##  $ avg_brunch_visits         : num  0 0 0 0 0 0 0 0 0 0 ...
```

```
##  $ avg_christmas_visits       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ avg_newyears_visits        : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ avg_valentine_visits       : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ avg_mothers_day_visits     : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ first_party_size           : int  2 8 6 2 6 4 4 4 2 3 ...
##  $ first_discount             : int  0 0 -28 0 0 -27 0 0 0 0 ...
##  $ first_spend                : num  115 436 375 100 188 ...
##  $ first_diffdays_madeon_visit: num  2.035 3.239 40.335 0.823 7.861 ...
##  $ first_diffmins_rsvp_open   : num  7.98 17.18 2.48 29.77 10.77 ...
##  $ first_weekday_visit        : int  0 1 0 1 1 1 0 1 1 0 ...
##  $ first_weekend_visit        : int  1 0 1 0 0 0 1 0 0 1 ...
##  $ first_breakfast            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ first_lunch_visit          : int  0 0 0 0 0 1 0 0 1 0 0 ...
##  $ first_dinner_visit         : int  1 1 1 1 0 1 1 0 1 1 ...
##  $ first_brunch_visit         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ first_christmas_visit      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ first_newyears_visit       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ first_valentine_visit      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ first_mother_day_flag      : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ num_of_different_servers   : int  1 1 1 1 1 1 1 1 4 1 ...
##  $ max_same_server            : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ num_of_different_tables    : int  1 1 1 1 1 1 1 1 4 1 ...
##  $ max_same_table             : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ avg_Beverage_spend         : num  8 44 17 0 4 0 22 15.5 0 10 ...
##  $ avg_Condiments_spend       : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ avg_Events_spend           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ avg_Food_spend             : num  0 212 255 72 204 -27 157 105 44 129 ...
##  $ avg_Liquor.Beer_spend      : num  16 112 0 0 0 0 0 0 7 0 ...
##  $ avg_Misc_spend             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ avg_Reserve.Wine_spend     : num  0 0 148 0 0 0 0 0 0 0 ...
##  $ avg_Retail_spend           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ avg_Wine_spend             : num  13 68 0 28 0 0 72 52 7 14 ...
##  $ avg_Beverage_qty           : num  2 7 5 0 1 0 4 4 0 1 ...
##  $ avg_Condiments_qty         : num  6 35 20 6 9 0 18 22 3.75 12 ...
##  $ avg_Events_qty             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ avg_Food_qty               : num  0 17 17 6 22 0 15 14 4 9 ...
##  $ avg_Liquor.Beer_qty        : num  1 9 0 0 0 0 0 0 0.5 0 ...
##  $ avg_Misc_qty               : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ avg_Reserve.Wine_qty       : num  0 0 2 0 0 0 0 0 0 0 ...
##  $ avg_Retail_qty             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ avg_Wine_qty               : num  1 5 0 2 0 0 2 4 0.5 1 ...
##  $ Anniversary_visits         : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Birthday_visits            : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ Other_visits               : int  1 1 1 1 1 1 1 1 4 1 ...
##  $ Cancelled_visits           : int  0 0 0 0 0 0 0 0 2 0 ...
##  $ No.Show_visits             : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ avg_days_bt_visits         : num  0 0 0 0 0 ...
##  $ same_day_visits            : int  0 0 0 0 0 0 0 0 0 0 ...
```

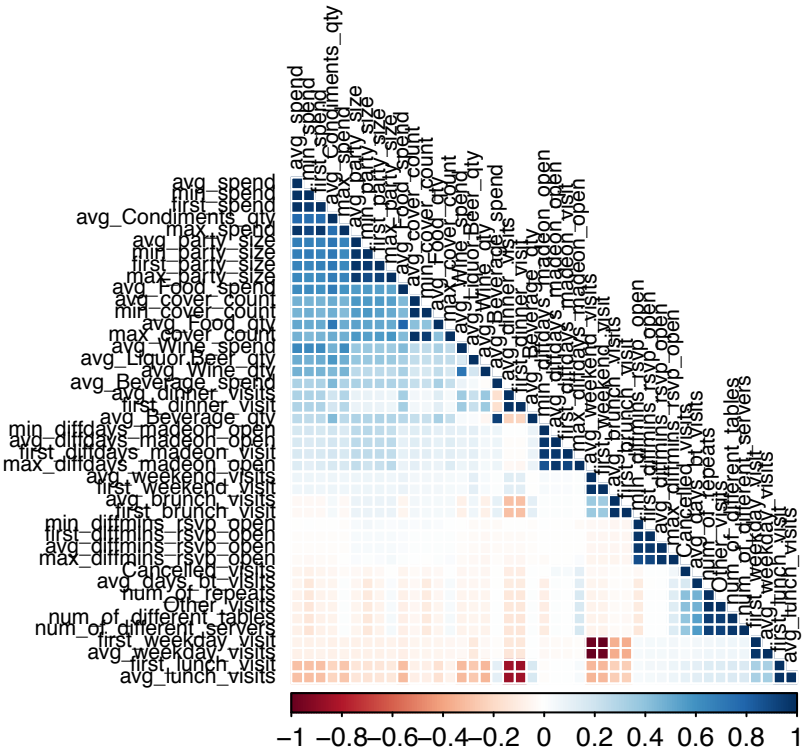
```
repeat.cluster <- as.data.frame(repeat.user)
repeat.numeric <- repeat.cluster[sapply(repeat.cluster,is.numeric)]

firsttime.cluster <- as.data.frame(firsttime.user)
firsttime.numeric <- firsttime.cluster[sapply(firsttime.cluster,is.numeric)]
```

```
rm(users.cluster,repeat.cluster,firsttime.cluster)

#keep for later merging
numeric.rownames <- data.frame(rows = row.names(user.numeric),loyalty_user_id = user$loyalty_user_id)
repeatnumeric.rownames <- data.frame(rows = row.names(repeat.numeric),loyalty_user_id = repeat.user$loya
firstnumeric.rownames <- data.frame(rows = row.names(firsttime.numeric),loyalty_user_id = firsttime.user

# Define functions used--------------------------------
#function for scaling
scale <- function(df){
  pre_range <- preProcess(df,method = c('center','scale'))
  processed <- predict(pre_range,df)
  return(data.frame(processed))
}

#graphs for within and between sum of squares
wss_and_bss <- function(df){
  #within sum of squares
  wss <- (nrow(df)-1)*sum(apply(df,2,var))
  for (i in 1:12) wss[i] <- sum(kmeans(df,
                                       centers=i)$withinss)
  print(plot(1:12, wss, type="b", xlab="Number of Clusters",
             ylab="Within groups sum of squares"))

  #between sum of squares
  bss <- (nrow(df)-1)*sum(apply(df,2,var))
  for (i in 1:12) bss[i] <- sum(kmeans(df,
                                       centers=i)$betweenss)
  print(plot(1:12, bss, type="b", xlab="Number of Clusters",
             ylab="Between groups sum of squares"))
}

#function will return the number of clusters (n) with loyalty user id for each cluster
#user is the scaled data frame, n is number of clusters, rows is the rownames+loyalty id to merge back

kmeans.venga <- function(user,n,rows){
  fit <- kmeans(user,n)
  #get cluster means:this illustrates amount of each characteristic in each cluster
  aggregate <- aggregate(user,by=list(fit$cluster), FUN=mean)
  #append cluster assignment
  cluster_assignment <- data.frame(rows = row.names(user),user, cluster_number = fit$cluster)
  #merge in loyalty user id; this gives us the user id by cluster that we can then merge into main data
  cluster_assignment.user <- merge(cluster_assignment,rows,by = 'rows')
  wss <- fit$withinss
  bss <- fit$betweenss
  list <- list(cluster_assignment.user,wss,bss,fit)
  return(list)
}

# I will first cluster all the data together, then repeat guests vs. non repeat

#1. All users : using user.numeric data set. ------------------------------------------------
```

```
#lot of variables, remove near zero variance and redundant (high collinearity vars), no NAs

set.seed(1234)
nzv <- nearZeroVar(user.numeric)
user.numeric <- user.numeric[, -nzv]
dim(user.numeric)
```

```
## [1] 50016    43
```

```
#look at correlation
userCor <- cor(user.numeric)
corrplot(userCor, order = "FPC", method = "color", type = "lower", tl.cex = 0.7, tl.col = rgb(0, 0, 0))
```




```
#remove highly correlated values
highlyCorrelated <- findCorrelation(userCor, cutoff=0.7)
highlyCorCol <- colnames(user.numeric)[highlyCorrelated]
user.numeric.clean <- user.numeric[,-which(colnames(user.numeric) %in% highlyCorCol)]

#scale the set; i will use user.numeric and user.numeric.clean (with highly correlated columns taken ou

#user.numeric - clustering
scaled.user.numeric <- scale(user.numeric)

#wss and bss plot to see optimal number of clusters
dev.off()
```

```
## null device
##           1
```

```
wss_and_bss(scaled.user.numeric)
```

```
## NULL
```

```
## NULL
```

```
#maybe 6, but optimal looks like 10


user.numeric.cluster <- kmeans.venga(scaled.user.numeric,6,numeric.rownames)
names <- c("user.cluster","user.wss","user.bss","user.fit")
names(user.numeric.cluster) <- names
names(user.numeric.cluster)
```

```
## [1] "user.cluster" "user.wss"     "user.bss"     "user.fit"
```

```
list2env(user.numeric.cluster,environment())
```

```
## <environment: R_GlobalEnv>
```

```
#build final data frame with all user observations and cluster assignments for each
user.final <- user.cluster %>% select(cluster_number,loyalty_user_id)
user.final <- merge(user,user.final,by = 'loyalty_user_id')

#number of observations in each cluster
table(user.cluster$cluster_number)
```

```
##
##     1     2     3     4     5     6
##  2162 11717 16125  4416    98 15498
```

```
##############DO NOT RUN, NEEDS TO BE FIXED#########################
#plot by cluster
#plotcluster(scaled.user.numeric.clean, user.fit$cluster)
#clusplot(userclean.cluster,user.fit$cluster, color = TRUE, shade = TRUE, labels = 2, lines = 0)
#only explain 38.19% variability, also a lot of variables. try and narrow down
####################################################################



scaled.user.numeric.clean <- scale(user.numeric.clean)

#using the cleaned user data with 15 variables
wss_and_bss(scaled.user.numeric.clean)
```

```
## NULL
```

```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 2500800)
```

```
## NULL
```

```
#7 clusters

user.numeric.clean <- kmeans.venga(scaled.user.numeric.clean,7,numeric.rownames)
names <- c("userclean.cluster","userclean.wss","userclean.bss")
names(user.numeric.clean) <- names
names(user.numeric.clean)
```

```
## [1] "userclean.cluster" "userclean.wss"     "userclean.bss"
## [4] NA
```

```
list2env(user.numeric.clean,environment())
```

```
## <environment: R_GlobalEnv>
```

```
table(userclean.cluster$cluster_number)
```

```
##
##     1     2     3     4     5     6     7
## 14640  3566 20181  3542  3497  2574  2016
```

```
#2. First time users ---------------------------------------------

set.seed(1234)
nzv <- nearZeroVar(firsttime.numeric)
firsttime.numeric <- firsttime.numeric[, -nzv]
dim(firsttime.numeric)
```

```
## [1] 42940    38
```

```
#look at correlation
firstuserCor <- cor(firsttime.numeric)
corrplot(firstuserCor, order = "FPC", method = "color", type = "lower", tl.cex = 0.7, tl.col = rgb(0, 0

#remove highly correlated values
highlyCorrelated.first <- findCorrelation(firstuserCor, cutoff=0.7)
highlyCorCol.first <- colnames(firsttime.numeric)[highlyCorrelated]
firsttime.numeric.clean <- firsttime.numeric[,-which(colnames(firsttime.numeric) %in% highlyCorCol)]

#scale the set; i will use firsttime.numeric.clean (with highly correlated columns taken out)

scaled.first.numeric <- scale(firsttime.numeric.clean)

dev.off()
```

```
## null device
##           1
```

```
wss_and_bss(scaled.user.numeric)
```

```
## NULL
```

```
## NULL
```

```
#8 clusters

firstuser.numeric.cluster <- kmeans.venga(scaled.first.numeric,8,firstnumeric.rownames)
names <- c("first.cluster","first.wss","first.bss","first.fit")
names(firstuser.numeric.cluster) <- names
names(firstuser.numeric.cluster)
```

```
## [1] "first.cluster" "first.wss"     "first.bss"     "first.fit"
```

```
list2env(firstuser.numeric.cluster,environment())
```

```
## <environment: R_GlobalEnv>
```

```
#build final data frame with all user observations and cluster assignments for each
firstuser.final <- first.cluster %>% select(cluster_number,loyalty_user_id)
firstuser.final <- merge(user,firstuser.final,by = 'loyalty_user_id')


#number of observations in each cluster
table(first.cluster$cluster_number)
```

```
##
##     1     2     3     4     5     6     7     8
##  1157    71  9580 13967  3326  3135  3066  8638
```

```
#interesting clusters to look at: 6- outliers, 4,7,2 - biggest groups

six <- firstuser.final %>% filter(cluster_number == 6)
#cluster six: all events

four <- firstuser.final %>% filter(cluster_number == 4)


#3. Repeat users ----------------------------

set.seed(1234)
nzv <- nearZeroVar(repeat.numeric)
repeat.numeric <- repeat.numeric[, -nzv]
dim(repeat.numeric)
```

```
## [1] 7076   46
```

```
#look at correlation
repeatuserCor <- cor(repeat.numeric)
corrplot(repeatuserCor, order = "FPC", method = "color", type = "lower", tl.cex = 0.7, tl.col = rgb(0, 

#remove highly correlated values
highlyCorrelated.repeat <- findCorrelation(repeatuserCor, cutoff=0.7)
highlyCorCol.repeat <- colnames(repeat.numeric)[highlyCorrelated]
```

```
repeat.numeric.clean <- repeat.numeric[,-which(colnames(repeat.numeric) %in% highlyCorCol)]

#scale the set; i will use firsttime.numeric.clean (with highly correlated columns taken out)

scaled.repeat.numeric <- scale(repeat.numeric.clean)

dev.off()
```

```
## null device
##           1
```

```
wss_and_bss(scaled.repeat.numeric)
```

```
## NULL
```

```
## NULL
```

```
#6 or 7 clusters

repeat.numeric.cluster <- kmeans.venga(scaled.repeat.numeric,6,repeatnumeric.rownames)
names <- c("repeat.cluster","repeat.wss","repeat.bss","repeat.fit")
names(repeat.numeric.cluster) <- names
names(repeat.numeric.cluster)
```

```
## [1] "repeat.cluster" "repeat.wss"     "repeat.bss"     "repeat.fit"
```

```
list2env(repeat.numeric.cluster,environment())
```

```
## <environment: R_GlobalEnv>
```

```
#number of observations in each cluster
table(repeat.cluster$cluster_number)
```

```
##
##    1    2    3    4    5    6
## 1253  610  264 2589  488 1872
```

```
#interesting clusters to look at:4- most obs


#build final data frame with all user observations and cluster assignments for each
repeat.final <- repeat.cluster %>% select(cluster_number,loyalty_user_id)
repeat.final <- merge(user,repeat.final,by = 'loyalty_user_id')

#4.Final analysis, on user,first and repeat final tables----------------------------

#only keep the final analysis tables
rm(list= ls()[!(ls() %in% c('repeat.final','firstuser.final','user','user.final'))])
```