City Search Tool - TD 2020

10/18/2020

User input

First, the user should enter two address of family members (or friends) that they would consider being close to as a factor that would influence their decision to move:

```
address1 <- "78 Tennis Villas Drive, CA, 92629"
address2 <- "24 Jupiter Hills Drive, CA, 92660"
address <- c(address1, address2)
```

Next, the user should enter an importance level for each factor. The importance level will be used to cluster the cities accordingly.

The rank scale goes from 1 to 5 with 5 being the most important, and 1 being the least important. (If all are equally important, give them all the same value.)

```
#rating by movehub (a moving company)
imp.mhr = 3
#purchase power
imp.pp = 3
#health care
imp.hc = 3
#pollution
imp.poll = 3
#quality of life
imp.qol = 3
#crime rating
imp.cr = 3
#parks and rec spending
imp.pr = 3
#entertainment and recreation (i.e. movies, restaurants, etc)
imp.er = 3
#nightlife
imp.nl = 3
#cost of nightlife
imp.cnl = 3
#sunshine
imp.sun = 3
#overall weather
imp.wthr = 3
#importance of distance from family 1
imp.f1 = 3
#importance of distance from family 2
imp.f2 = 3
```

####Importing the Data

not for user use

```
#Set working Directory
setwd("/Users/carolineogden/Documents/Texas A&M University/Datathon2020")
#Load in data
Data.us <- read.csv("DataFinal.csv", header=TRUE)
```

Computing distance from family

not for user use

```
#Computing the latitude and longitude of the addresses of family members
addresses <- tribble( ~addr,
                      address[1],
                      address[2])
coordinates <- addresses %>%
     geocode(addr)
coordinates
```

```
## # A tibble: 2 x 3
##   addr                                lat  long
##   <chr>                             <dbl> <dbl>
## 1 78 Tennis Villas Drive, CA, 92629  33.5 -118.
## 2 24 Jupiter Hills Drive, CA, 92660  33.6 -118.
```

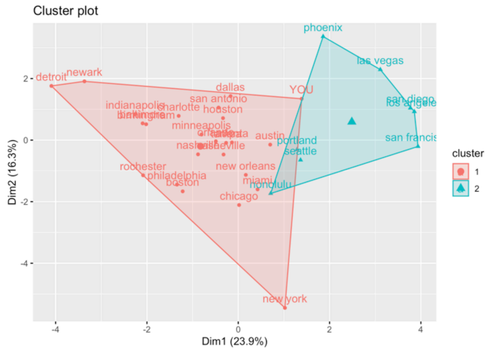
```
#Computing the distance in miles between each city and the family members house
get_geo_distance = function(long1, lat1, long2, lat2, units = "miles") {
  loadNamespace("purrr")
  loadNamespace("geosphere")
  longlat1 = purrr::map2(long1, lat1, function(x,y) c(x,y))
  longlat2 = purrr::map2(long2, lat2, function(x,y) c(x,y))
  distance_list = purrr::map2(longlat1, longlat2, function(x,y) geosphere::distHaversine(x, y))
  distance_m = list.extract(distance_list, position = 1)
  if (units == "km") {
    distance = distance_m / 1000.0;
  }
  else if (units == "miles") {
    distance = distance_m / 1609.344
  }
  else {
    distance = distance_m
    # This will return in meter as same way as distHaversine function.
  }
  distance
}

#Merging the original dataset with the distance (in miles) from both family members
n=dim(Data.us)[1]
Data.us$family1 = c(rep(0, n))
Data.us$family2 = c(rep(0, n))
for (i in 1:n) {
  Data.us$family1[i] <- get_geo_distance(coordinates[1,3], coordinates[1,2], Data.us[i,'lng'], Data.us[i,'lat'],
units = "miles")
  Data.us$family2[i] <- get_geo_distance(coordinates[2,3], coordinates[2,2], Data.us[i,'lng'], Data.us[i,'lat'],
units = "miles")
}
#Data.us
```

Standardize all features, then multiply by the weight given by the user

```
Data.us[is.na(Data.us)] = 0
rownames(Data.us) = Data.us$City


#Data.us$lat = scale(Data.us$lat)
#Data.us$lng = scale(Data.us$lng)
Data.us$Movehub.Rating = scale(Data.us$Movehub.Rating) * imp.mhr
Data.us$Purchase.Power = scale(Data.us$Purchase.Power) * imp.pp
Data.us$Health.Care = scale(Data.us$Health.Care) * imp.hc
Data.us$Pollution = scale(Data.us$Pollution) * -imp.poll
Data.us$Quality.of.Life = scale(Data.us$Quality.of.Life) * imp.qol
Data.us$Crime.Rating = scale(Data.us$Crime.Rating) * -imp.cr
Data.us$ParksandRecSpending = scale(Data.us$ParksandRecSpending) * imp.pr
Data.us$EnternainmentandRecRank = scale(Data.us$EnternainmentandRecRank) * imp.er
Data.us$NightlifeRank = scale(Data.us$NightlifeRank) * imp.nl
Data.us$NightlifeCostRank = scale(Data.us$NightlifeCostRank) * imp.cnl
Data.us$SunshineHoursPerYear = scale(Data.us$SunshineHoursPerYear) * imp.sun
Data.us$CamelotClimateIndex = scale(Data.us$CamelotClimateIndex) * imp.wthr
Data.us$family1 = scale(Data.us$family1) * -imp.f1
Data.us$family2 = scale(Data.us$family2) * -imp.f2

#add a new data point to show where the user fits best
YOU = c(imp.mhr, imp.pp, imp.hc, imp.poll, imp.qol, imp.cr, imp.pr, imp.er, imp.nl, imp.cnl, imp.sun, imp.wthr, i
mp.f1, imp.f2)

Data.us = rbind(Data.us, YOU)
rownames(Data.us)[32] = "YOU"
Data.us2 = Data.us[,-(1:2)]
rownames(Data.us2)[32] = "YOU"
```

Cluster with this “weighted” data set

```
ch.index = function(x,kmax,iter.max=100,nstart=10,algorithm="Lloyd")
{
  ch = numeric(length=kmax-1)
  n = nrow(x)
  for (k in 2:kmax) {
    a = kmeans(x,k,iter.max=iter.max,nstart=nstart, algorithm=algorithm)
    w = a$tot.withinss
    b = a$betweenss
    ch[k-1] = (b/(k-1))/(w/(n-k))
}
return(list(k=2:kmax,ch=ch)) }

CH = ch.index(Data.us2, kmax=10)
bestk = CH$k[which.min(CH$ch)]

k5 = kmeans(Data.us2, centers = bestk, nstart = 25)
fviz_cluster(k5, data=Data.us2)
```



The idea is that the user would find the point labeled “YOU” on the chart. The recommended cities to move to are located within the same cluster as the “YOU” point. They can also determine if they want to go outside of their “comfort zone”/“ideal locations” by looking at other clusters and the cities within those.