HEOM Distance Metric

The HEOM Metric

For measuring the distance between 2 organizations, we are using a Heterogeneous Euclidean Overlap Distance Metric for measuring the distance between 2 organizations. See equation 1 of the following paper: https://projecteuclid.org/journals/missouri-journal-of-mathematical-sciences/volume-22/issue-2/Heterogeneous-Distance-Measures-and-Nearest-Neighbor-Classification-in-an-Ecological/10.35834/mjms/1312233141.full

The distance metric is as follows:

Say there are m characteristics to compare and n comparison orgs. Let $\boldsymbol{x}$ be the vector characteristics of the input organization and $\boldsymbol{y}_j$ be the vector characteristics of the comparison organization.

For each organization j, for i in 1, ... , m:

$$HEOM_i(x_i, y_{j,i}) = \begin{cases} \frac{|x_i - y_{j,i}|}{range\{y_{j,i}\}} & \text{if characteristic i is numeric} \\ s(x_i, y_{j,i}) & \text{if characteristic i is state} \\ p(x_i, y_{j,i}) & \text{if characteristic i is categrtorical that is not state} \end{cases}$$

$$s(x_i, y_{j,i}) = \begin{cases} 0 & \text{if input organization and the comparison org are in the same state} \\ 1/3 & \text{if input organization and the comparison org are in the same cencus division} \\ 2/3 & \text{if input organization and the comparison org are in neighbooring cencus regions} \\ 1 & otherwise \end{cases}$$

$$p(x_i, y_{j,i}) = \begin{cases} 0 & \text{if } x_i = y_{j,i} \\ \frac{\#(y_{j,i} \in \{\text{ soft match critera}\})}{n} & \text{if soft matching} \\ 1 & \text{otherwise} \end{cases}$$

Then,

$$HEOM(\boldsymbol{x}, \boldsymbol{y}_j) = \frac{1}{m} \sum_{i=1}^{m} HEOM_i(x_i, y_{j,i})$$

The census regions can be found here : https://www2.census.gov/geo/pdfs/maps-data/maps/reference/us_regdiv.pdf. We combined census region 1 and 2 to just be one region.

Example

We will walk though comparing and input organization to one org in the comparison set. (In the HEOM_with_weights function it is exactly this process just with a for loop over all orgs in the comparison set).

Say we have the following set up (see pretend-shiny vignette for a walk through of this set up):

```
source("global.R") #needed for state distance
#org characteristics
org <- list(State = "DC",
            Loc = "Metropolitan",
            MajorGroup = 2,
            NTEE = "B",
            NTEE.CC = NA,
            UNIV = FALSE,
            HOSP = FALSE,
            TotalExpense = 3000000,
            TotalEmployee = 9
)

#search criteria
search <- list(State = c("VA", "DC", "MD"),
               MajorGroup = 2,
               Loc = c("Metropolitan"),
               NTEE = c("B"),
               NTEE.CC = NA,
               UNIV = 1,
               HOSP = 1,
               TotalExpense = c(1000000, 6000000),
               TotalEmployee = c(0, 100)
)

#TRUE is hard match, FALSE is soft match
#TotalExpense and TotalEmployee do not have hard/soft options
hard <- list(State = FALSE,
             Loc = TRUE,
             MajorGroup = TRUE,
             NTEE = TRUE,
             NTEE.CC = FALSE,
             TotalExpense = TRUE,
             TotalEmployee = TRUE
)

dat.filtered <- dat_filtering_hard(search, hard)
```

We first transform all numerical variables to the log scale and set the needed parameters.

```
dat.filtered <- dat.filtered%>%
  dplyr::mutate(dist = NA) %>%
  dplyr::mutate(log.expense = log(TotalExpense+1, base = 10)) %>%
  dplyr::mutate(log.employee = log(TotalEmployee+1, base = 10))

#needed values
A <- 9 #number of attributes to compare
n <- 1 # = dim(dat.filtered)[1] in real setting, just set =1 for example
D <- matrix(0, nrow = n, ncol = A)

colnames(D) <- c("logTotalExpense", "TotalEmployee", "State", "MajorGroup", "NTEE", "NTEE.CC", "UNIV",

#get categorical comparisons tables for probabilities
```

```
# State, Major Group, NTEE, NTEE.CC, UNIV, HOSP, and Location type
# d <- 0 if org i attribute == org attribute
# d <- p matching else if org i attribute %in% search criteria
# d <- 1 o.w.

tab.majorgroup <- base::table(dat.filtered$MajorGroup) / n
tab.ntee <- base::table(dat.filtered$NTEE) / n
tab.ntee.cc <- base::table(dat.filtered$NTEE.CC) / n
tab.univ <- base::table(dat.filtered$UNIV) / n
tab.hosp <- base::table(dat.filtered$HOSP) / n
tab.loc <- base::table(dat.filtered$LocationType) / n

#Get weights for categorical variables,
#weight = 0 if there was a hard matching on only one option
#weight = 0 if exact match with multiple hard matching options, = 1 if no match with multiple hard math

#weight.state <- sum(tab.state[which(names(tab.state) %in% search$State)])
weight.majorgroup <- sum(tab.majorgroup[which(names(tab.majorgroup) %in% search$MajorGroup)])
weight.ntee <- sum(tab.ntee[which(names(tab.ntee) %in% search$NTEE)])
#weight.ntee.cc <- sum(tab.ntee.cc[which(names(tab.ntee.cc) %in% search$NTEE.CC)])
weight.loc <- sum(tab.loc[which(names(tab.loc) %in% search$Loc)])

  #if anything is NA assign that distance 1
```

Then we calculate the distance for each attribute (scaled between 0 and 1):

```
#just one org to compare to for this example so just set i=1
#in the actual function you do for i in 1:(dim(dat.filtered)[1])
i=1

#Total Expense
if(is.na(dat.filtered$TotalExpense[i])){
  D[i, "logTotalExpense"] <- 1
}else{
  r.log.expense <- max(dat.filtered$log.expense) -  min(dat.filtered$log.expense)   #range
      D[i, "logTotalExpense"] <- abs(dat.filtered$log.expense[i] - log(org$TotalExpense, 10)) / r.log.e
    }

#Total Employee
## distance for total employee
if(is.na(dat.filtered$TotalEmployee[i])){
  D[i, "TotalEmployee"] <- 1
}else{
  r.log.employee <- max(dat.filtered$log.employee) -  min(dat.filtered$log.employee) #range
      D[i, "TotalEmployee"] <- abs(dat.filtered$log.employee[i] - log(org$TotalEmployee+1, 10)) / r.log
    }
# Next do categorical comparisons
#State has its own function
if(is.na(dat.filtered$State[i])){
  D[i, "State"] <- 1
}else{
  D[i, "State"] <- state_distance(org$State, dat.filtered$State[i])
}
```

```
# Every Categorical but state uses proportions as weights
if(is.na(dat.filtered$MajorGroup[i])){
  D[i, "MajorGroup"] <- 1
}else{
  D[i, "MajorGroup"] <- ifelse(dat.filtered$MajorGroup[i] == org$MajorGroup, 0,
                               ifelse(dat.filtered$MajorGroup[i] %in% search$MajorGroup, weight.majorgrou
}

if(is.na(dat.filtered$NTEE[i])){
  D[i, "NTEE"] <- 1
}else{
  D[i, "NTEE"] <- ifelse(dat.filtered$NTEE[i] == org$NTEE, 0,
                         ifelse(dat.filtered$NTEE[i] %in% search$NTEE, weight.ntee, 1 ))
    }

if(is.na(dat.filtered$LocationType[i] )){
  D[i, "Loc"] <- 1
}else{
  D[i, "Loc"] <-  ifelse(dat.filtered$LocationType[i] == org$Loc, 0,
                         ifelse(dat.filtered$LocationType[i] %in% search$Loc, weight.loc, 1 ))
}

if(is.na(dat.filtered$UNIV[i] )){
  D[i, "UNIV"] <- 1
}else{
  D[i, "UNIV"] <- ifelse(dat.filtered$UNIV[i] == org$UNIV, 0 , 1)
}

if(is.na(dat.filtered$HOSP[i] )){
  D[i, "HOSP"] <- 1
}else{
  D[i, "HOSP"] <- ifelse(dat.filtered$HOSP[i] == org$HOSP, 0 , 1)
}
```

We then find the mean of all of these attribute distances to get the overall distance. (Use a weighted mean for weighting some attributes more than others. See section below.)

```
distance <- rowSums(D) / A #normalize by number of matched attributes.
```

Why state calculation is different than the other categorical variables

State has a different distance calculation than the other categorical variables. This is because state categories are Likert-Scale-esk as in just because state A $\neq$ state B and state A $\neq$ state C does not mean that state A is the same mathematical distance from state B as it is from state C because of the states have physical locations. This is different than the other categorical variables, since if org A is in broad category I, org B is in broad category II, and org C is in broad category III, the org A IS the same mathematical distance from state B as it is from state C because the orders of the broad category have no physical meaning. But taking into account physical distance (in miles or km) is also not representative of how "similar" state are to each other because the state have difference sizes and difference densities throughout the US. So settle these issues we have implemented the following solution:

The basic idea is that, if two orgs are in the same state they have a distance of 0, if they are in difference states but the same region they have a distance of 1/3, if they are in different but neighboring regions, they have a

distance of 2/3, and if they are non-neighboring regions they have a distance of 1. See funcs/state-distance.R for the code for implementing this distance.

We use the census divisions as our regions except we combine New England and Middle Atlantic into one region. https://www2.census.gov/geo/pdfs/maps-data/maps/reference/us_regdiv.pdf

Here is a table of each region, states in the region, neighboring regions, and non-neighboring regions:

Region Number	**Name of Region**	**State in the Region**	**Neighboring Regions**	**Non-neighboring regions**
1	New England and Middle Atlantic:	Connecticut, Maine, Massachusetts, New Hampshire, Rhode Island, Vermont, New Jersey, New York, Pennsylvania, Delaware	3, 5	4, 6, 7, 8, 9
3	East North Central	Indiana, Illinois, Michigan, Ohio, Wisconsin	1, 4, 6	5, 7, 8, 9
4	West North Central	Iowa, Kansas, Minnesota, Missouri, Nebraska, North Dakota, South Dakota	3, 7 ,8	1, 4, 5, 6, 9
5	South Atlantic	District of Columbia, Florida, Georgia, Maryland, North Carolina, South Carolina, Virginia, West Virginia, Puerto Rico	1, 3, 6	2, 4, 5, 7, 8, 9
6	East South Central	Alabama, Kentucky, Mississippi, Tennessee	3, 5, 7	1, 4, 6, 8, 9
7	West South Central	Arkansas, Louisiana, Oklahoma, Texas	4, 6, 8	1, 3, 5, 7, 9
8	Mountain	Arizona, Colorado, Idaho, New Mexico, Montana, Utah, Nevada, Wyoming	4, 7, 9	1, 3, 5, 6, 8
9	Pacific	Alaska, California, Hawaii, Oregon, Washington	8	1, 3, 4, 5, 6, 7

First, put this in the global.R file

```
region <- data.table::data.table(state = sort(c(state.abb, "PR", "DC")),
                                 region = c(9, 6, 7, 8, 9, 8, 1, 5, 1, 5,
                                            5, 9, 5, 8, 3, 3, 4, 6, 7, 1,
                                            1, 1, 6, 4, 4, 6, 8, 5, 4, 4,
                                            1, 1, 8, 8, 1, 3, 7, 9, 1, 5,
                                            1, 5, 4, 6, 7, 8, 5, 1, 8, 3,
                                            5, 8))
```

The, the state distance metric is implemented in the funcs/state-distance.R file. The function is a series of if statements with the following check structure.

1. if search.state == NA then return 1
2. if org.state == search.state then return 0
3. else if org.region == search.region then return 1/3
4. else if search.region neighbors org.region (according to the above table) then return 2/3
5. else return 1

How to add/delete attributes to compare

Common code is not currently implemented in this distance metric. This is something we would like to do in the future, but conversations need to be had about the best way of doing this (ie. do some common codes need to be grouped together? or we just not consider some common codes because there are so few, or no, orgs in the population with those codes)

Adding or deleting attributes to compare on is a very simple task. For adding a new attribute, simply follow the format for either numerical or categorical variables based on what type of attribute you want to add, change `A` to reflect the number of attributes you are comparing on, get either the rage for a numerical or the table of probabilities for a categorical, and add an additional if/if else statement as the forms above for the actual attribute distance.

To delete an attribute, just delete that if/if else statement then change `A` to reflect the number of attributes you are comparing on.

How to add weights to the distance calculation

If you want to add weights, follow this procedure.

Say we have A attributes labeled 1, 2. ..., A. Assign weights a_1, a_2, ..., a_A such that $\sum_i a_i = 1$. Then the weighed distances is $\sum_i a_i d_i$. The unweighted averages is when $a_i = 1/A$ for all i.

```
a <- c(1, 3, 3, 4, 1, 1, 1, 1, 3) / 18
weighted.average <- sum(a*D)
weighted.average
```

```
## [1] 0.1883135
```