ACMT Examples: Calculating Neighborhood Composite Measures

Amy Youngbloom

3/22/2022

Introduction

Researchers looking at the effect of neighborhood on various health outcomes have utilized a variety of composite measures. Such measures look across several socioeconomic, housing, and demographic measures to operationalize measures such as neighborhood deprivation, vulnerability, and fragmentation. Many of these measures use American Community Survey variables in constructing the composite measure and many of these variables are built into the default ACS variable list in the ACMT. As such, pulling ACS variables and building an estimated composite measures for given locations can be easily done with the ACMT. Below is code for how to pull the relevant variables and construct several such measures using the ACMT. Note that the code is set up to pull variable from the 2019 5-year American Community survey. For other surveys, see the ACS Variables by year document.

For each composite measures, a geocoded list of addresses must first be created. For these examples, we will use a list of Seattle High School and Middle Schools as our locations and examine a 1000 m radius around each school.

Create a list of schools and addresses

```
library(tidyverse)
library(janitor)
acsvars<-read_csv('ACMT/ACSColumns.csv')
source("GeocoderACMT.R")

#Create list of schools and addresses

seattle.schools<-c('Alan T. Sugiyama High School', 'Ballard High School', 'Chief Sealth Internat
ional High School', 'Cleveland High School', 'Franklin High School', 'Garfield High School', 'In
graham High School', 'Interagency Academy', 'Lincoln High School', 'Nathan Hale High School', 'N
ova', 'Ranier Beach High School', 'Roosevelt High School', 'Seattle World School', 'Skills Cente
r', 'The Center School', 'West Seattle High School', 'Aki Kurose Middle School', 'Denny Internat
ional Middle School', 'Eckstein Middle School', 'Hamilton International Middle School', 'Jane Ad
dams Middle School', 'Madison Middle School', 'McClure Middle School', 'Meany Middle School', 'M
ercer International Middle School', 'Robert Eagle Staff Middle School', 'Washington Middlw Schoo
l', 'Whitman Middle School')

seattle.address<-c('8601 Rainier Ave. S, Seattle, WA 98118','1418 NW 65th St., Seattle, WA 9811
7', '2600 SW Thistle St., Seattle, WA 98126', '5511 15th Ave. S, Seattle, WA 98108', '3013 S Mt
 Baker Blvd., Seattle, WA 98144', '400 23rd Ave., Seattle, WA 98122', '1819 N 135th St., Seattl
e, WA 98133', '3528 S Ferdinand St., Seattle, WA 98118', '4400 Interlake Ave. N, Seattle, WA 981
03', '10750 30th Ave. NE, Seattle, WA 98125', '2410 E Cherry St., Seattle, WA 98122', '8815 Sewa
rd Park Ave. S, Seattle, WA 98118', '1410 NE 66th St., Seattle, WA 98115', '1700 E Union St., Se
attle, WA 98122', '2445 3rd Ave. S., Seattle, WA 98134', '305 Harrison St., Seattle, WA 98109',
'3000 California Ave. SW, Seattle, WA 98116', '3928 S Graham St, Seattle, WA 98118', '2601 SW Ke
nyon St, Seattle, WA, 98126', '3003 NE 75th St, Seattle, WA 98115', '1610 N 41st St, Seattle, WA
98103', '11051 34th Ave NE, Seattle, WA 98125', '3429 45th Ave SW, Seattle, WA 98116', '1915 1st
Ave W, Seattle, WA 98119', '301 21st Ave E, Seattle, WA 98112', '1600 S Columbian Way, Seattle,
 WA, 98108', '1330 N 90th St, Seattle, WA 98103', '2101 S Jackson St, Seattle, WA 98144', '9201
 15th Ave NW, Seattle, WA 98117')

schools<-data.frame(seattle.schools)
schools$address<-seattle.address
```

Geocode Addresses

```
#create lat and long columns
schools<-schools %>%
  mutate(lat=NA,
         long=NA)

#Geocoding loop
for (i in 1:nrow(schools)) {
  if(!is.na(schools$lat[i])) next #skip already geocoded
  if(is.na(schools$address[i])) next #skip NA address values
  print(i) #print the number to track progress

  address<-schools$address
  lat_long<-geocode(address[i])
  schools$lat[i]<-lat_long$latitude # add latitude to dataset
  schools$long[i] <-lat_long$longitude # add longitude to dataset
}
```

```
head(schools)
```

```
##                          seattle.schools
## 1           Alan T. Sugiyama High School
## 2                    Ballard High School
## 3 Chief Sealth International High School
## 4                  Cleveland High School
## 5                   Franklin High School
## 6                   Garfield High School
##                                    address      lat      long
## 1   8601 Rainier Ave. S, Seattle, WA 98118 47.52617 -122.2701
## 2      1418 NW 65th St., Seattle, WA 98117 47.67608 -122.3750
## 3   2600 SW Thistle St., Seattle, WA 98126 47.52837 -122.3660
## 4      5511 15th Ave. S, Seattle, WA 98108 47.55337 -122.3136
## 5 3013 S Mt Baker Blvd., Seattle, WA 98144 48.16430 -122.4806
## 6         400 23rd Ave., Seattle, WA 98122 47.60505 -122.3025
```

Pulling ACS Variables for Neighborhood Composite Measures

Variables for the following neighborhood-level measures are built into the list of ACS variables in the ACMT:

1. Area Deprivation Index (Singh, 2003)
2. Condogen's Social Fragmentation Index (Congdon, 2013)
3. Social Vulnerability Index (Flanagan, 2011)

---

1. Area Deprivation Index

Singh's composite measures of deprivation provides a validated measure looking across socioeconomic variables. This factor-based index is calculated by multiplying the value by the factor coefficients for each of the 17 variables below. The measure is standardized around a mean of 100, with a standard deviation of 20.

To contruct this index, we first set the list of relevant variables.

```
acs.2019<-tidycensus::load_variables(2019, 'acs5', cache=TRUE)
adi_variables<-c('B15003_002', #no school completed
                 'B15003_003', #nursery school completed
                 'B15003_004', #kindergarten
                 'B15003_005', #1st grade
                 'B15003_006', #2nd grade
                 'B15003_007', #3rd grade
                 'B15003_008', #4th grade
                 'B15003_009', #5th grade
                 'B15003_010', #6th grade
                 'B15003_011', #7th grade
                 'B15003_012', #8th grade
                 'B15003_013', #9th grade
                 'B15003_017', #HS Graduate
                 'B15003_018', #GED or other degree
                 'B15003_019', #some college, less than 1 year
                 'B15003_020', # some college 1 year or more, no degree
                 'B15003_021', #associates degree
                 'B15003_022', #bachelors degree
                 'B15003_023', #masters degree
                 'B15003_024', #professional degree
                 'B15003_025', #doctoral degree
                 'B15003_001', #Total population 25 and over
                 'C24030_018', #males in professional scientific and technical services
                 'C24030_019', #males in management
                 'C24030_045', #females in professional, scientific and technical services
                 'C24030_046', #females in management
                 'C24030_002', #males, 16 and older, employed
                 'C24030_029', #females, 16 and older, employed
                 'B19013_001', #median household income
                 'B19001_002', #less than 10k
                 'B19001_011', # $50-$59.9k
                 'B19001_012', # $60-$74.9k
                 'B19001_013', # $75k-$99.9k
                 'B19001_014', # $100k-$124.9k
                 'B19001_015', # $125k-$149.9k
                 'B19001_016', # $150k - $199.9k
                 'B19001_017', # $200k or more
                 'B25077_001', #median home value
                 'B25064_001', #median gross rent
                 'B25088_002', #monthly owner costs, housing units with a mortgage
                 'B25003_002', #owner-occupied units
                 'B25003_001', #total occupied housing uits
                 'B23025_005', #unemployed
                 'B23025_001', #Total 16 years and older
                 'B17023_002', #Total households in poverty
                 'B17023_001', #Total households, povery determined
                 'B06012_002', #below 100% povery level
                 'B06012_003', #100 to 149% of poverty level
                 'B06012_001', #households, poverty level determined
                 'B11003_016', #female householder, no spouse or partner, with own children unde
r 18
```

```
                  'B11003_010', #male householder, no spouse or partner, with own children under
 18
                  'B11003_001', #total families
                  'B25044_003', #total owner-occupied, no vehicle
                  'B25044_010', #renter-occupied, no vehicle
                  'B25044_001', #total, vehicle determined
                  'B25043_001', #total, telephone determined
                  'B25043_007', #owner-occupied, no telephone
                  'B25043_016', #renter occupied, no telephone
                  'B25016_007', #owner occupied, lacking complete plumbing
                  'B25016_016', #renter occupied, lacking complete plumbing
                  'B25016_001', #households, plumbing determined
                  'B25014_005', #owner occupied 1.01 to 1.5 per room
                  'B25014_006', #owner occupied 1.51 to 2 per room
                  'B25014_007', #owner occupied 2.01 or more per room
                  'B25014_011', #renter occupied 1.01 to 1.5 per room
                  'B25014_012', #renter occupied 1.51 to 2 per room
                  'B25014_013', #renter occupied 2.01 or more per room
                  'B25014_001') #total occupied units
```

Next we ust the list of variables to add columns for each variable to the dataset (counts and proportions)

```
#Identify the variables of interest from the default list of ACS variables

acsvars<-read_csv('ACMT/ACSColumns.csv')
acsvars<-subset(acsvars, acs_col %in% adi_variables)

##create 'count' versions of each variable name and 'proportion' versions for each #ACS variable
where applicable
acs_count_names<-paste(acsvars$var_name, "count", sep="_")
if (length(acsvars$var_name[acsvars$universe_col != ""]) == 0) {   # prevent having something th
at is exactly "_proportion"
  acs_proportion_names <- character(0)
} else {
  acs_proportion_names <- paste(acsvars$var_name[!is.na(acsvars$universe_col)], "proportion", se
p="_")   # only non-universal variables have proportions
}

#Set the list of variable codes, the list of variable names, the radius, and the year for the da
ta you want pulled
codes_of_acs_variables_to_get<-acsvars$acs_col
names_of_variable_to_get<-c(acs_count_names, acs_proportion_names)
radius <- c(1000)#set the radius for the area of interest
year <- 2019 #set the year for the data of interest

#add columns to dataset to add variables to
var.cols<-data.frame(matrix(nrow=nrow(schools), ncol=length(names_of_variable_to_get))) #create
 dataset of columns
colnames(var.cols)<-names_of_variable_to_get #name the columns
schools_adi<-cbind(var.cols, schools) #bind the columns to the dataset
```

Once the columns have been added, we can run a loop to use the ACMT to pull each variable

```
#run loop to pull variables
for(address in 1:nrow(schools_adi)) {
   tryCatch({if(!is.na(schools_adi[,1][address])) next #skip the row if the data is already there
  if(!is.na(schools_adi[,1][address])) next #skip the row if the data is already there
  print(address) #print the number to keep track of progress
  latitude<-schools_adi$lat[address] #set lat
  longitude<-schools_adi$long[address] #set long

  environmental_measures<-get_acmt_standard_array(long=longitude, lat=latitude, radius_meters = radius, year=year, codes_of_acs_variables_to_get = codes_of_acs_variables_to_get) #pull measures for given lat & long

      for(name_of_variable in names_of_variable_to_get){ #for each measures, get the value and put it into the column of the same name
     value_of_variable <- environmental_measures[environmental_measures$names == name_of_variable, ]$values
     schools_adi[[name_of_variable]][address]<-value_of_variable
  }

 for (name_of_variable in names_of_variable_to_get) {
        schools_adi[[name_of_variable]][address] <- environmental_measures[environmental_measures$names == name_of_variable, ]$values
 }},error=function(e){cat("ERROR :", conditionMessage(e), "\n")}) #this will print any error messages
}
```

Now we can calucate the ADI measures from the ACS variables that were pulled.

```
adi_measure_dataset <- schools_adi %>%
  mutate(less_than_9yrs_education= ((no_education_count +
                                       pre_school_count + kindergarten_count +
                                       first_grade_count + second_grade_count +
                                       third_grade_count + fourth_grade_count +
                                       fifth_grade_count + sixth_grade_count +
                                       seventh_grade_count + eighth_grade_count +
                                       ninth_grade_count) / pop_25_and_over_count),
         hs_education_or_more = (high_school_grad_count +
                                    ged_or_alt_diploma_count + some_college_less_than_1_year_cou
nt +
                                    some_college_1_year_or_more_count + associates_degree_count
+
                                    bachelors_degree_count + masters_degree_count +
                                    professional_degree_count + doctoral_degree_count)/
            pop_25_and_over_count,
         white_collar_occ=(males_in_professional_occup_count +
                             males_in_management_count +
                             females_in_professional_occup_count +
                             females_in_management_count) / males_16_older_workforce_count,
         income_disparity = (100*(hhincome_less_than_10k_count/
                                    (hhincome_50k_to_59k_count + hhincome_60k_to_74k_count +
                                       hhincome_75k_to_99k_count + hhincome_100k_to_124k_count +
                                       hhincome_125k_to_149k_count + hhincome_150k_to_199k_count
+
                                       hhincome_200k_or_more_count))),
                                  below_150_poverty = (pop_below_100_poverty_threshold_count +
                                                         pop_100_to_149_poverty_threshold_coun
t)/ total_pop_poverty_count,
                                  single_parent_with_kids = (female_head_kids_count +
                                                               male_head_kids_count)/total_famil
ies_count,
                                  hh_novehicle = (owner_no_vehicle_count +
                                                    renter_no_vehicle_count)/ occupied_housing_v
ehicle_determined_count,
                                  hh_no_phone = (no_phone_owner_count + no_phone_renter_count) /
                                    total_occupied_housing_units_tele_count,
                                  hh_no_plumb = (no_comp_plumb_owner_count+no_comp_plumb_renter_
count)/
                                    total_occupied_housing_units_plumb_count,
                                  hh_crowded=(owner_1.01_to_1.5_per_room_count +
                                                owner_1.51_to_2.0_per_room_count +
                                                owner_2.01_or_more_per_room_count +
                                                renter_1.01_to_1.5_per_room_count +
                                                renter_1.51_to_2.0_per_room_count +
                                                renter_2.01_or_more_per_room_count)/
                                    total_occupied_housing_units_room_count)
```

Next we subset the dataset to just the school namd and address and the variables that will be included in the principal component analysis.

```
schools_adi$fam_below_poverty_proportion
```

```
##  [1] 0.11743768 0.02415449 0.07095482 0.06998068 0.03776713 0.08367946
##  [7] 0.04955272 0.13335622 0.02442616 0.06027300 0.07276888 0.12989587
## [13] 0.04703083 0.07963853 0.00000000 0.02413345 0.02833876 0.15814287
## [19] 0.08252235 0.02678554 0.02341664 0.06253947 0.02738762 0.02441850
## [25] 0.06098892 0.11092015 0.03014743 0.10494608 0.03021027
```

```
adi_measures<-adi_measure_dataset %>%
  dplyr::select(seattle.schools, address, less_than_9yrs_education, hs_education_or_more, white_collar_occ, med_hincome_count, income_disparity, med_home_val_count, median_rent_count, median_mortgage_count, owner_occupied_units_proportion, unemployed_proportion, fam_below_poverty_proportion, below_150_poverty, single_parent_with_kids, hh_novehicle, hh_no_phone, hh_no_plumb, hh_crowded)

head(adi_measures)
```

```
##                         seattle.schools
## 1           Alan T. Sugiyama High School
## 2                    Ballard High School
## 3 Chief Sealth International High School
## 4                  Cleveland High School
## 5                   Franklin High School
## 6                   Garfield High School
##                                   address less_than_9yrs_education
## 1   8601 Rainier Ave. S, Seattle, WA 98118              0.118666274
## 2      1418 NW 65th St., Seattle, WA 98117              0.006873042
## 3   2600 SW Thistle St., Seattle, WA 98126              0.054669713
## 4      5511 15th Ave. S, Seattle, WA 98108              0.127070954
## 5 3013 S Mt Baker Blvd., Seattle, WA 98144              0.004019281
## 6         400 23rd Ave., Seattle, WA 98122              0.037015242
##   hs_education_or_more white_collar_occ med_hincome_count income_disparity
## 1            0.8469905        0.1254256          70470.39        14.829432
## 2            0.9815984        0.4315409         114119.82         2.652467
## 3            0.8985922        0.1969071          82014.46         5.569553
## 4            0.8323459        0.2087176          83957.09         5.154960
## 5            0.9777540        0.1481489          77494.15         3.702490
## 6            0.9411624        0.4177231         105820.89         9.588673
##   med_home_val_count median_rent_count median_mortgage_count
## 1           418610.7          1373.703              2125.790
## 2           694207.1          1822.459              2744.724
## 3           439726.2          1467.869              2164.156
## 4           475737.5          1596.075              2102.733
## 5           351838.6          1147.995              1851.580
## 6           673774.0          1703.574              2693.619
##   owner_occupied_units_proportion unemployed_proportion
## 1                       0.5565090            0.04122008
## 2                       0.4616595            0.03648610
## 3                       0.5498524            0.03112392
## 4                       0.6581606            0.03751051
## 5                       0.9307744            0.02265127
## 6                       0.4278077            0.03474142
##   fam_below_poverty_proportion below_150_poverty single_parent_with_kids
## 1                   0.11743768         1.0579897               0.1355521
## 2                   0.02415449         0.3316639               0.0682368
## 3                   0.07095482         0.7008092               0.1320667
## 4                   0.06998068         0.5692136               0.1132606
## 5                   0.03776713         0.3574029               0.0600289
## 6                   0.08367946         1.1308565               0.1254846
##   hh_novehicle hh_no_phone hh_no_plumb hh_crowded
## 1   0.13488614 0.019340998 0.005705783 0.10136853
## 2   0.08988524 0.022533471 0.003624895 0.04421752
## 3   0.08892802 0.009818168 0.004759712 0.02578555
## 4   0.08787650 0.017696377 0.001285120 0.05057544
## 5   0.01769702 0.010906289 0.003403343 0.02201620
## 6   0.17946655 0.020910723 0.004181691 0.03152043
```

Now we run the PCA, excluding the school name address variables (1st and 2nd columns in the dataset)

```
adi_pca<-stats::princomp(na.omit(adi_measures[3:19]), cor=TRUE)
summary(adi_pca)
```

```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3     Comp.4     Comp.5
## Standard deviation     2.9610583 1.9198780 1.0629339 0.93076386 0.85504476
## Proportion of Variance 0.5157568 0.2168195 0.0664605 0.05096008 0.04300597
## Cumulative Proportion  0.5157568 0.7325763 0.7990368 0.84999692 0.89300290
##                            Comp.6     Comp.7     Comp.8     Comp.9     Comp.10
## Standard deviation     0.70053909 0.58751537 0.54560948 0.46021435 0.401332167
## Proportion of Variance 0.02886794 0.02030437 0.01751116 0.01245866 0.009474559
## Cumulative Proportion  0.92187084 0.94217521 0.95968637 0.97214503 0.981619590
##                            Comp.11     Comp.12     Comp.13     Comp.14
## Standard deviation     0.335610694 0.286524733 0.263170839 0.168697650
## Proportion of Variance 0.006625561 0.004829201 0.004074052 0.001674053
## Cumulative Proportion  0.988245151 0.993074353 0.997148405 0.998822458
##                             Comp.15      Comp.16      Comp.17
## Standard deviation     0.1150079944 0.0754937482 3.304650e-02
## Proportion of Variance 0.0007780493 0.0003352533 6.423947e-05
## Cumulative Proportion  0.9996005072 0.9999357605 1.000000e+00
```

For factor 1, each variable has a given loading, which we can look at with the following code:

```
adi_pca$loadings[,1]
```

```
##        less_than_9yrs_education            hs_education_or_more
##                     0.300374152                    -0.307398959
##                white_collar_occ                med_hincome_count
##                    -0.241355143                    -0.304972277
##                income_disparity               med_home_val_count
##                     0.284862265                    -0.252885909
##               median_rent_count            median_mortgage_count
##                    -0.244168143                    -0.270323049
## owner_occupied_units_proportion            unemployed_proportion
##                    -0.005446497                     0.256957058
##    fam_below_poverty_proportion                below_150_poverty
##                     0.295063978                     0.225719257
##          single_parent_with_kids                     hh_novehicle
##                     0.223835214                     0.063130302
##                     hh_no_phone                      hh_no_plumb
##                     0.193572344                     0.139576864
##                      hh_crowded
##                     0.266687659
```

Using the PCA factor loadings, we can calculate a weighted Area Deprivation measure by multiplying each value by the factor loading for each variable, adding the values to create a composite measures, and standardizing the composite measure.

```
adi_pca$loadings[17]
##assign loading values for each variable
less_than_9yrs_loading<-adi_pca$loadings[1]
hs_education_or_more_loading<-adi_pca$loadings[2]
white_collar_occ_loading<-adi_pca$loadings[3]
med_hincome_count_loading<-adi_pca$loadings[4]
income_disparity_loading<-adi_pca$loadings[5]
med_home_val_count_loading<-adi_pca$loadings[6]
median_rent_count_loading<-adi_pca$loadings[7]
median_mortgage_count_loading<-adi_pca$loadings[8]
owner_occupied_units_proportion_loading<-adi_pca$loadings[9]
unemployed_proportion_loading<-adi_pca$loadings[10]
fam_below_poverty_proportion_loading<-adi_pca$loadings[11]
below_150_poverty_loading<-adi_pca$loadings[12]
single_parent_with_kids_loading<-adi_pca$loadings[13]
hh_novehicle_loading<-adi_pca$loadings[14]
hh_no_phone_loading<-adi_pca$loadings[15]
hh_no_plumb_loading<-adi_pca$loadings[16]
hh_crowded_loading<-adi_pca$loadings[17]

#Calculated & standardize weighted NDI value using pca loadings
adi_measures <-adi_measures%>%
  mutate(adi_value=(less_than_9yrs_loading*less_than_9yrs_education)+
           (hs_education_or_more_loading*hs_education_or_more)+
           (white_collar_occ_loading*white_collar_occ)+
           (income_disparity_loading*income_disparity)+
           (below_150_poverty_loading*below_150_poverty)+
           (single_parent_with_kids_loading*single_parent_with_kids)+
           (hh_novehicle_loading*hh_novehicle)+
           (hh_no_phone_loading*hh_no_phone)+
           (hh_no_plumb_loading*hh_no_plumb)+
           (hh_crowded_loading*hh_crowded)+
           (med_hincome_count_loading*med_hincome_count)+
           (med_home_val_count_loading*med_home_val_count)+
           (unemployed_proportion_loading*unemployed_proportion)+
           (owner_occupied_units_proportion_loading*owner_occupied_units_proportion)
                    *-1) %>%
  mutate(adi_standardized=(adi_value-mean(adi_value))/sd(adi_value))
```

The standardized Area Deprivation Index can be used to compare neighborhoods around each Seattle high school.

```
summary(adi_measures$adi_standardized)
```

```
adi_mean_table<-adi_measures %>%
  group_by(seattle.schools) %>%
  summarise_at(vars(adi_standardized), list(adi_mean=mean))%>%
  arrange(adi_mean)

adi_mean_table
```

2. Congdon’s Social Fragmentation Index (Congdon, 2013)

Social fragmentation describes an ecological measure of community integration and has been studied primarily in association with mental health and suicidality. Social fragmentation is operationalized using measures of single adults living along, the proportion of residents who moved into an area recently (i.e., in the last 5 years), and proportion of renters and vacancies in the area. These measures can all be pulled from American Community Survey data variables that are built into the ACMT. Below we walk through the code to pull the relevant variables and construct the social fragmentation index.

We first designate the list of relevant variables that we will be pulling, pull those variables from the default list of American Community Survey variables and create proportion and count names.

```
sfi_variables<-c('B25003_001', #total occupied units
                 'B25003_002', #owner-occupied units
                 'B25002_003', #Total vacant units
                 'B25002_001', #total units, vacancy determined
                 'B11011_001', #total households
                 'B11010_010', #female householder, living alone
                 'B11010_003', #male householder, living alone
                 'B25129_003', #owner, moved in <5 years ago
                 'B25129_039', #renter moved in <5 years ago
                 'B25129_001') #total households

#Identify the variables of interest from the default list of ACS variables
acsvars<-read_csv('ACMT/ACSColumns.csv')
acsvars<-subset(acsvars, acs_col %in% sfi_variables)

##create 'count' versions of each variable name and 'proportion' versions for each #ACS variable
where applicable
acs_count_names<-paste(acsvars$var_name, "count", sep="_")
if (length(acsvars$var_name[acsvars$universe_col != ""]) == 0) {   # prevent having something th
at is exactly "_proportion"
  acs_proportion_names <- character(0)
} else {
  acs_proportion_names <- paste(acsvars$var_name[!is.na(acsvars$universe_col)], "proportion", se
p="_")   # only non-universal variables have proportions
}

#Set the list of variable codes, the list of variable names, the radius, and the year for the da
ta you want pulled
codes_of_acs_variables_to_get<-acsvars$acs_col
names_of_variable_to_get<-c(acs_count_names, acs_proportion_names)
radius<- c(1000)#set the radius for the area of interest
year <- 2019 #set the year for the data of interest

var.cols<-data.frame(matrix(nrow=nrow(schools), ncol=length(names_of_variable_to_get))) #create
 dataset of columns
colnames(var.cols)<-names_of_variable_to_get #name the columns
schools_sfi<-cbind(var.cols, schools) #bind the columns to the dataset
```

Next, we can set the list of variable codes and names, and set the radius and year of the data you are interested in. Once these values are set, we can run the ACMT to pull the measures for each geocoded address.

```
#run loop to pull variables
for(address in 1:nrow(schools_sfi)) {
   tryCatch({if(!is.na(schools_sfi[,1][address])) next #skip the row if the data is already ther
e
  if(!is.na(schools_sfi[,1][address])) next #skip the row if the data is already there
  print(address) #print the number to keep track of progress
  latitude<-schools_sfi$lat[address] #set lat
  longitude<-schools_sfi$long[address] #set long

  environmental_measures<-get_acmt_standard_array(long=longitude, lat=latitude, radius_meters =
 radius, year=year, codes_of_acs_variables_to_get = codes_of_acs_variables_to_get) #pull measure
s for given lat & long

      for(name_of_variable in names_of_variable_to_get){ #for each measures, get the value and p
ut it into the column of the same name
     value_of_variable <- environmental_measures[environmental_measures$names == name_of_variabl
e, ]$values
     schools_sfi[[name_of_variable]][address]<-value_of_variable
  }

 for (name_of_variable in names_of_variable_to_get) {
        schools_sfi[[name_of_variable]][address] <- environmental_measures[environmental_measure
s$names == name_of_variable, ]$values
 }},error=function(e){cat("ERROR :", conditionMessage(e), "\n")}) #this will print any error mes
sages
}
```

Once the data is pulled for each variables, we can calculate the SFI value

```
sfi_measures<-schools_sfi %>%
  mutate(housing_not_owner_occupied = (occupied_units_count - owner_occupied_units_count)/occupi
ed_units_count,
         living_alone = (female_householder_alone_count + male_householder_alone_count)/total_ho
useholds_count,
         recent_move = (renter_occupied_recent_move_count+owner_occupied_recent_move_count)/tota
l_occupied_housing_year_count) %>%
  mutate(housing_not_owner_occupied_stand = (housing_not_owner_occupied-mean(housing_not_owner_o
ccupied))/sd(housing_not_owner_occupied),
         vacant_housing_stand = (vacant_units_proportion-mean(vacant_units_proportion))/sd(vacan
t_units_proportion),
         living_alone_stand = (living_alone-mean(living_alone))/sd(living_alone),
         recent_move_stand = (recent_move-mean(recent_move))/sd(recent_move)) %>%
           mutate(sfi_value = housing_not_owner_occupied_stand + vacant_housing_stand + living_a
lone_stand + recent_move_stand)

summary(sfi_measures$sfi_value)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
## -4.3691 -1.9199 -0.3698  0.0000  1.6183  7.1316
```

Now that the standardized social fragmentation index has been calculated for each location in the address list, values can be used for additional analyses.

```
summary(sfi_measures$sfi_value)
```


```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
## -4.3691 -1.9199 -0.3698  0.0000  1.6183  7.1316
```


```
sfi_mean_table<-sfi_measures %>%
  group_by(address) %>%
  summarise_at(vars(sfi_value), list(sfi_mean=mean))

sfi_mean_table
```

```
## # A tibble: 29 x 2
##    address                                  sfi_mean
##    <chr>                                       <dbl>
##  1 10750 30th Ave. NE, Seattle, WA 98125     -3.14  
##  2 11051 34th Ave NE, Seattle, WA 98125      -3.30  
##  3 1330 N 90th St, Seattle, WA 98103         -0.143 
##  4 1410 NE 66th St., Seattle, WA 98115        1.31  
##  5 1418 NW 65th St., Seattle, WA 98117       -0.370 
##  6 1600 S Columbian Way, Seattle, WA, 98108  -1.75  
##  7 1610 N 41st St, Seattle, WA 98103          3.40  
##  8 1700 E Union St., Seattle, WA 98122        5.49  
##  9 1819 N 135th St., Seattle, WA 98133        0.0673
## 10 1915 1st Ave W, Seattle, WA 98119          3.21  
## # … with 19 more rows
```

3. CDC Social Vulnerability Index

The Social Vulnerability Index (CITE) is used by the CDC to rank census tracts according to their ability to prevent suffering in loss during disasters. This can be used in emergency preparedness efforts to identify potentially vulnerable areas, and can assist in estimating levels of supplies and support a community may need in their recovery efforts. Measures included in this index span across social and economic factors, including poverty, education, employment, income, ages, disabilities, household type, minority status, language, and housing and transportation. Variables are used to contruct percentile rankings across 4 domains: (1) Socioeconomic, (2) Household Composition, (3) Minority Status/Language, and (4) Housing Type/Transportation.

Other Social Vulnerability measures have also been developed and used in research on both disaster and emergency preparedness and in health prevention and promotion. Variables used in the CDC's Social Vulnerability Index are included in the default list of ACS variables, and thus can be easily pulled to construct the index for your research.

We'll start by designating the variables that we will need for the index.

```
svi_variables<-c('B01001_001', #Total Population
                 'B25001_001', #Total housing units
                 'B11003_001', #Total families
                 'B17001_002', #households below poverty
                 'B17001_001', #Total households, poverty determined
                 'B23025_001', #Residents 16 and older
                 'B23025_005', #Unemployed
                 'B19301_001', #per capital income
                 'B06009_002', #no high school diploma
                 'B06009_001', #Resident 25 and older
                 'B01001_020', #(Male!!65 and 66 years)
                 'B01001_021', #(Male!!67 to 69 years)
                  'B01001_022', #(Male!!70 to 74 years)
                  'B01001_023', #(Male!!75 to 79 years)
                  'B01001_024', #(Male!!80 to 84 years)
                  'B01001_025', #(Male!!85 years and over)
                  'B01001_044', #(Female!!65 and 66 years)
                  'B01001_045', #(Female!!67 to 69 years)
                  'B01001_046', #(Female!!70 to 74 years)
                  'B01001_047', #(Female!!75 to 79 years)
                  'B01001_048', #(Female!!80 to 84 years)
                  'B01001_049', #(Female!!85 years and over)
                 'B01001_003', #males under 5
                 'B01001_004', #males 5 to 9
                 'B01001_005', #males 10 to 14
                 'B01001_006', #Males 15 to 17
                 'B01001_027', #females under 5
                 'B01001_028', #females 5 to 9
                 'B01001_029', #females 10 to 14
                 'B01001_030', #females 15 to 17
                 'B18101_004', #males under 5 with a disability
                 'B18101_007', #males 5 to 17 with a disability
                 'B18101_010', #males 18 to 34, with a disability
                 'B18101_013', #males, 35 to 64, with a disability
                 'B18101_016', #males, 65 to 74, with a disability
                 'B18101_019', #males 75 years and over, with a disability
                 'B18101_023', #female under 5, with a disability
                 'B18101_026', #female 5 to 17, with a disability
                 'B18101_029', #female 18 to 34, with a disability
                 'B18101_032', #female 35 to 64, with a disability
                 'B18101_035', #female 65 to 74, with a disability
                 'B18101_038', #females 75 and older, with a disability
                 'B18101_001', #total pop, disability determined
                 'B11003_010', # male householder, no wife, own children under 18
                 'B11003_016', #female householder, no husband, children under 18
                 'B01001H_001', #non-Hispanic White
                 'B16005_007', #(native born, spanish speaker, speak english 'not well')
                  'B16005_008',  #(native born, Spanish speaker, speaks english, 'not at all'
                  'B16005_012',  #(native born, speak other indo-european language, speaks Engli
sh 'not well')
                  'B16005_013', #(native born, Speaks other indo-european language, speaks Engli
sh 'not at all')
```

```
                  'B16005_017', #native born, Asian & PI languages, speaks english 'not well'
                  'B16005_018', #native born, Asian & PI languages, speaks English 'not at all'
                  'B16005_022', #native born, speaks other languages, speaks English 'not well'
                  'B16005_023',  #native born, speaks other languages, speaks Enlgish 'not at al
l'
                  'B16005_029', #foreign born, speaks Spanish, speaks English, 'not well'
                  'B16005_030', #foreign born, speaks Spanish, speaks English, 'not at all'
                  'B16005_034', #foreign born, speaks other indo-European language, speaks Engli
sh 'not well'
                  'B16005_035',  #foreign born, speaks other Indo-European language, speaks Engl
ish 'not at all'
                  'B16005_039', #foreign born, speaks Asian & PI languages, speaks English 'not
 well'
                  'B16005_040', #foreign born, speaks Asian & PI languages, speaks English 'not
 at all'
                  'B16005_044', #foreign born, speaks other languages, speaks English 'not well'
                  'B16005_045', #foreign born, speaks other languages, speaks English, 'not at a
ll'
                 'B16005_001', #total population, 5 years and older
                 'B25024_007', #10 to 19 units
                 'B25024_008', #20 to 49 units
                 'B25024_009', #50 units or more
                 'B25024_010', #mobile homes
                 'B25024_001', #total housing units
                 'B25014_005', #owner occupied, 1.01 to 1.5 units
                 'B25014_006', #owner occupied, 1.51 to 2 units
                 'B25014_007', #owner occupied, 2.01 units or more
                 'B25014_011', #renter occpuied, 1.01 to 1.5 units
                 'B25014_012', #renter occupied, 1.51 to 2 units
                 'B25014_013', #renter occpuied, 2.01 or more units
                 'B25014_001', #total units
                 'B25044_003', #owner occupied, no vehicle
                 'B25044_010', #renter occupied, no vehicle
                 'B25044_001', #total occupied units, vehicles determined
                 'B26001_001') #total in group quarters
```

Next add a column for each variable to be pulled to the dataset.

```
#Identify the variables of interest from the default list of ACS variables
acsvars<-read_csv('ACMT/ACSColumns.csv')
acsvars<-subset(acsvars, acs_col %in% svi_variables)

##create 'count' versions of each variable name and 'proportion' versions for each #ACS variable where applicable
acs_count_names<-paste(acsvars$var_name, "count", sep="_")
if (length(acsvars$var_name[acsvars$universe_col != ""]) == 0) {   # prevent having something that is exactly "_proportion"
  acs_proportion_names <- character(0)
} else {
  acs_proportion_names <- paste(acsvars$var_name[!is.na(acsvars$universe_col)], "proportion", sep="_")   # only non-universal variables have proportions
}
```

Once the variables are designated, we can create a column for each variable to be pulled and set the radius and year of the data that will be pulled.

```
#Set the list of variable codes, the list of variable names, the radius, and the year for the data you want pulled
codes_of_acs_variables_to_get<-acsvars$acs_col
names_of_variable_to_get<-c(acs_count_names, acs_proportion_names)
radius <- c(1000)#set the radius for the area of interest
year <- 2019 #set the year for the data of interest

#create a dataset of columns for each variable to be pulled
var.cols<-data.frame(matrix(nrow=nrow(schools), ncol=length(names_of_variable_to_get))) #create dataset of columns
colnames(var.cols)<-names_of_variable_to_get #name the columns
schools_svi<-cbind(var.cols, schools) #bind the columns to the dataset
```

Now we can run a loop to pull the variables for each location

```
#run loop to pull variables
for(address in 1:nrow(schools_svi)) {
   tryCatch({if(!is.na(schools_svi[,1][address])) next #skip the row if the data is already ther
e
  if(!is.na(schools_svi[,1][address])) next #skip the row if the data is already there
  print(address) #print the number to keep track of progress
  latitude<-schools_svi$lat[address] #set lat
  longitude<-schools_svi$long[address] #set long

  environmental_measures<-get_acmt_standard_array(long=longitude, lat=latitude, radius_meters =
 radius, year=year, codes_of_acs_variables_to_get = codes_of_acs_variables_to_get) #pull measure
s for given lat & long

      for(name_of_variable in names_of_variable_to_get){ #for each measures, get the value and p
ut it into the column of the same name
     value_of_variable <- environmental_measures[environmental_measures$names == name_of_variabl
e, ]$values
     schools_svi[[name_of_variable]][address]<-value_of_variable
  }

 for (name_of_variable in names_of_variable_to_get) {
        schools_svi[[name_of_variable]][address] <- environmental_measures[environmental_measure
s$names == name_of_variable, ]$values
 }},error=function(e){cat("ERROR :", conditionMessage(e), "\n")}) #this will print any error mes
sages
}
```

Next we can combine values and variables as needed to create the final list of measures which aligns with the Social Vulnerability Index. These measures include margins of error for each variable, for which a standard of 90% is used (based on the Census Bureau MOE standards).

```
#function to calculate 90% MOE
alpha=.10
deg.free=(length(schools)-1)
t.score<-qt(p=alpha/2, df=deg.free, lower.tail = F)
std_moe<-function(x) t.score*(sd(x)/sqrt(length(x)))

#calcualte estimates & margins of errors for relevant variables
svi_measures<-schools_svi %>%
  #First create the estimates
  mutate(e_totpop=total_pop_count,
         e_hu=housing_units_count,
         e_hh=total_families_count,
         e_pov=below_pov_count,
         e_unemp=unemployed_count,
         e_pci=per_capita_income_count,
         e_nohsdp=no_hsdiploma_count,
         e_age65=(males_65_to_66_count + males_67_to_69_count +
                    males_70_to_74_count+ males_75_to_79_count+
                    males_80_to_84_count+ males_85_and_older_count+
                    females_65_to_66_count + females_67_to_69_count+
                    females_70_to_74_count + females_75_to_79_count+
                    females_80_to_84_count + females_85_and_older_count),
         e_age17=(males_under_5_count+males_5_to_9_count+males_10_to_14_count+males_15_to_17_cou
nt+    females_under_5_count+females_5_to_9_count+females_10_to_14_count+females_15_to_17_coun
t),
         e_disabl=(males_under5_disability_count+males_5_17_disability_count+males_18_34_disabil
ity_count+males_35_64_disability_count+males_65_74_disability_count+males_75_plus_disability_cou
nt+females_under5_disability_count+females_5_17_disability_count+females_18_34_disability_count+
females_35_64_disability_count+females_65_74_disability_count+females_75_plus_disability_count),
         e_sngpnt=(male_head_kids_count+female_head_kids_count),
         e_minrty=(total_pop_count-non_hisp_white_count),
         e_limeng=(ntv_sp_lmt_eng_notwell_count+ntv_sp_lmt_eng_notatall_count+ntv_ie_lmt_eng_not
well_count+
                              ntv_ie_lmt_eng_notatall_count+ntv_api_lmt_en_notwell_count+ntv_api
_lmt_en_notatall_count+
                              ntv_oth_lmt_en_notwell_count+ntv_oth_lmt_en_notatall_count+fb_sp_l
mt_eng_notwell_count+
                              fb_sp_lmt_eng_notatall_count+fb_ie_lmt_eng_notwell_count+fb_ie_lmt
_eng_notatall_count+
                              fb_api_lmt_en_notwell_count+fb_api_lmt_en_notatall_count+fb_oth_lm
t_en_notwell_count+
                              fb_oth_lmt_en_notatall_count),
         e_munit=(units_10_to_19_count+units_20_to_49_count+units_50_ormore_count)/total_housing
_units_count_in_structure_count,
         e_mobile=mobile_homes_count,
          e_crowd=(owner_1.01_to_1.5_per_room_count+owner_1.51_to_2.0_per_room_count+owner_2.01_
or_more_per_room_count+renter_1.01_to_1.5_per_room_count+renter_1.51_to_2.0_per_room_count+rente
r_2.01_or_more_per_room_count)/total_occupied_housing_units_room_count,
         e_noveh=(owner_no_vehicle_count+renter_no_vehicle_count)/occupied_housing_vehicle_deter
mined_count,
         e_groupq=group_quarters_count) %>%
  #next calculate the margins of errors for each estimate
```

```
mutate(m_totpop=std_moe(e_totpop),
       m_hu=std_moe(e_hu),
       m_hh=std_moe(e_hh),
       m_pov=std_moe(e_pov),
       m_unemp=std_moe(e_unemp),
       m_pci=std_moe(e_pci),
       m_nohsdp=std_moe(e_nohsdp),
       m_age65=std_moe(e_age65),
       m_age17=std_moe(e_age17),
       m_disabl=std_moe(e_disabl),
       m_sngpnt=std_moe(e_sngpnt),
       m_minrty=std_moe(e_minrty),
       m_limeng=std_moe(e_limeng),
       m_munit=std_moe(e_munit),
       m_mobile=std_moe(e_mobile),
       m_crowd=std_moe(e_crowd),
       m_noveh=std_moe(e_noveh),
       m_groupq=std_moe(e_groupq)) %>%
#next calculate the percentages for each variable
mutate(ep_pov=below_pov_proportion,
       ep_unemp=unemployed_proportion*100,
       ep_pci=per_capita_income_count*100,
       ep_nohsdp=no_hsdiploma_proportion*100,
       ep_age65=(e_age65/e_totpop)*100,
       ep_age17=(e_age17/e_totpop)*100,
       ep_disabl=(e_disabl/civilian_pop_count)*100,
       ep_sngpnt=(e_sngpnt/total_families_count)*100,
       ep_minrty=(e_minrty/e_totpop)*100,
       ep_limeng=(e_limeng/total_ages_5_up_count)*100,
       ep_munit=(e_munit/e_hu)*100,
       ep_mobile=mobile_homes_proportion*100,
       ep_crowd=(e_crowd/total_occupied_housing_units_room_count)*100,
       ep_noveh=(e_noveh/occupied_housing_vehicle_determined_count)*100,
       ep_groupq=(e_groupq/e_totpop)*100) %>%
#Next calculate the margin of error for the percentages of each variable
mutate(mp_pov=std_moe(ep_pov),
       mp_unemp=std_moe(ep_unemp),
       mp_pci=std_moe(ep_pci),
       mp_nohsdp=std_moe(ep_nohsdp),
       mp_age65=std_moe(ep_age65),
       mp_age17=std_moe(ep_age17),
       mp_disabl=std_moe(ep_disabl),
       mp_sngpnt=std_moe(ep_sngpnt),
       mp_minrty=std_moe(ep_minrty),
       mp_limeng=std_moe(ep_limeng),
       mp_munit=std_moe(ep_munit),
       mp_mobile=std_moe(ep_mobile),
       mp_crowd=std_moe(ep_crowd),
       mp_noveh=std_moe(ep_noveh),
       mp_groupq=std_moe(ep_groupq)
) %>%
#Next calculate the percentiles for each variable
```

```
  mutate(epl_unemp=percent_rank(ep_unemp),
         epl_pci=percent_rank(ep_pci),
         epl_pov=percent_rank(ep_pov),
         epl_nohsdp=percent_rank(ep_nohsdp),
         epl_age65=percent_rank(ep_age65),
         epl_age17=percent_rank(ep_age17),
         epl_disabl=percent_rank(ep_disabl),
         epl_sngpnt=percent_rank(ep_sngpnt),
         epl_minrty=percent_rank(ep_minrty),
         epl_limeng=percent_rank(ep_limeng),
         epl_munit=percent_rank(ep_munit),
         epl_mobile=percent_rank(ep_mobile),
         epl_crowd=percent_rank(ep_crowd),
         epl_noveh=percent_rank(ep_noveh),
         epl_groupq=percent_rank(ep_groupq)
         )%>%
    #Next we calculate the 4 theme variables by summing the percentile ranking for each variable
in a given theme
    mutate(spl_theme1=(epl_pov+epl_unemp+epl_pci+epl_nohsdp),
           spl_theme2=(epl_age65+epl_age17+epl_disabl+epl_sngpnt),
           spl_theme3=(epl_minrty+epl_limeng),
           spl_theme4=(epl_munit+epl_mobile+epl_crowd+epl_noveh+epl_groupq)
           ) %>%
  #Next we calulate the percentile rank for each of the 4 theme varialbes
    mutate(rpl_theme1=percent_rank(spl_theme1),
           rpl_theme2=percent_rank(spl_theme2),
           rpl_theme3=percent_rank(spl_theme3),
           rpl_theme4=percent_rank(spl_theme4)) %>%
#Next, we sum theme variables and calculate the percentiles
  mutate(spl_themes=spl_theme1+spl_theme2+spl_theme3+spl_theme4) %>%
  mutate(rpl_themes=percent_rank(spl_themes)) %>%
  #next we can calculate flags for values in the the 90th percentile
  mutate(f_pov=ifelse(epl_pov>=.90, 1, 0),
         f_unemp=ifelse(epl_unemp>=.90, 1, 0),
         f_pci=ifelse(epl_pci>=.9, 1, 0),
         f_nohsdp=ifelse(epl_nohsdp>=.9, 1, 0),
         f_age65=ifelse(epl_age65>=.9, 1, 0),
         f_age17=ifelse(epl_age17>=.9, 1, 0),
         f_disabl=ifelse(epl_disabl>=.9,1,0),
         f_sngpnt=ifelse(epl_sngpnt>=.9,1,0),
         f_minrty=ifelse(epl_minrty>=.9,1,0),
         f_limeng=ifelse(epl_limeng>=.9,1,0),
         f_munit=ifelse(epl_munit>=.9,1,0),
         f_mobile=ifelse(epl_mobile>=.9,1,0),
         f_crowd=ifelse(epl_crowd>=.9,1,0),
         f_noveh=ifelse(epl_noveh>=.9,1,0),
         f_groupq=ifelse(epl_groupq>=.9,1,0)) %>%
  #Sum the flags for each theme variable
  mutate(f_theme1=f_pov+f_unemp+f_pci+f_nohsdp,
         f_theme2=f_age65+f_age17+f_disabl+f_sngpnt,
         f_theme3=f_minrty+f_limeng,
         f_theme4=f_munit+f_mobile+f_crowd+f_noveh+f_groupq) %>%
```

```
  #sum the flags
  mutate(f_total=f_theme1+f_theme2+f_theme3+f_theme4)
```

The total number of flags calculated for each location can be used to compare the raeas around each high school.

```
summary(svi_measures$f_total)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   0.000   0.000   1.000   1.552   2.000   6.000
```

```
svi_mean_table<-svi_measures %>%
  group_by(address)%>%
  summarise_at(vars(f_total), list(svi_mean=mean))

svi_mean_table
```

```
## # A tibble: 29 x 2
##    address                                  svi_mean
##    <chr>                                       <dbl>
##  1 10750 30th Ave. NE, Seattle, WA 98125           0
##  2 11051 34th Ave NE, Seattle, WA 98125            0
##  3 1330 N 90th St, Seattle, WA 98103               0
##  4 1410 NE 66th St., Seattle, WA 98115             1
##  5 1418 NW 65th St., Seattle, WA 98117             0
##  6 1600 S Columbian Way, Seattle, WA, 98108        6
##  7 1610 N 41st St, Seattle, WA 98103               0
##  8 1700 E Union St., Seattle, WA 98122             1
##  9 1819 N 135th St., Seattle, WA 98133             4
## 10 1915 1st Ave W, Seattle, WA 98119               1
## # … with 19 more rows
```

References

Singh, G. K. (2003). Area deprivation and widening inequalities in US mortality, 1969–1998. American journal of public health, 93(7), 1137-1143.

Congdon P. Assessing the impact of socioeconomic variables on small area variations in suicide outcomes in England. Int J Environ Res Public Health. 2013;10(1):158–77.

Flanagan, B. E., Gregory, E. W., Hallisey, E. J., Heitgerd, J. L., & Lewis, B. (2011). A social vulnerability index for disaster management. Journal of homeland security and emergency management, 8(1).