ACMT Examples: Calculating Neighborhood Composite Measures

Amy Youngbloom

1/12/2022

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.

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

```
source("setup-acmt.R")
library(tidyverse)
library(janitor)

#Create list of schools and addresses

seattle.schools<-list('Alan T. Sugiyama High School', 'Ballard High School', 'Chief Sealth International

seattle.address<-list('8601 Rainier Ave. S, Seattle, WA 98118','1418 NW 65th St., Seattle, WA 98117', '1

address_list<-structure(seattle.address, names=(seattle.schools))

##geocode address list
address_to_lat_long_list<-lapply(address_list, geocode)
```

```
print(address_to_lat_long_list[1])
```

```
## $`Alan T. Sugiyama High School`
## $`Alan T. Sugiyama High School`$latitude
## [1] 47.52617
##
## $`Alan T. Sugiyama High School`$longitude
## [1] -122.2701
```

Each composite measure also requires you to write the function to utilize the ACMT. See the code below to review this step

```
## Create the function to get the environmental measures:
get_variable_measures_from_acmt <- function(address_to_lat_long_list, radius_vector, year, names_of_vari
  address_to_radius_to_variable_to_measures_list <- vector(mode="list", length=length(address_to_lat_lo
  names(address_to_radius_to_variable_to_measures_list) <- names(address_to_lat_long_list)

  for(address in names(address_to_radius_to_variable_to_measures_list)) {
    radius_to_variable_to_measures_list <- vector(mode="list", length=length(radius_vector))
```

```
    names(radius_to_variable_to_measures_list) <- as.character(radius_vector)
    for (radius in radius_vector) {
      print(address)
      print(radius)

      latitude <- address_to_lat_long_list[[address]]$latitude
      longitude <- address_to_lat_long_list[[address]]$longitude

      environmental_measures <- get_acmt_standard_array(long=longitude,
                                                        lat=latitude, radius_meters = radius, year=year

      variable_to_measures_list <- vector(mode="list", length=length(names_of_variable_to_get))
      names(variable_to_measures_list) <- names_of_variable_to_get
      for (name_of_variable in names_of_variable_to_get) {
        value_of_variable <- environmental_measures[environmental_measures$names == name_of_variable, ]
        variable_to_measures_list[[name_of_variable]] <- value_of_variable
      }
      radius_to_variable_to_measures_list[[which(radius == radius_vector)]] <- variable_to_measures_list
      radius_to_variable_to_measures_list[[as.character(radius)]] <- variable_to_measures_list
    }
    address_to_radius_to_variable_to_measures_list[[address]] <- radius_to_variable_to_measures_list
  }
  return(address_to_radius_to_variable_to_measures_list)
}
```

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.

```
adi_variables<-c('B15003_002', 'B15003_003', 'B15003_004', 'B15003_005', 'B15003_006', 'B15003_007', 'B
```

Next we identify the relevant variables out of the full default list, and set the names and codes of the variables to pull using the ACMT function.

```
#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
```

```
acs_count_names<-paste(acsvars$var_name, "count", sep="_")
if (length(acsvars$var_name[acsvars$universe_col != ""]) == 0) {   # prevent having something that is e
  acs_proportion_names <- character(0)
} else {
  acs_proportion_names <- paste(acsvars$var_name[!is.na(acsvars$universe_col)], "proportion", sep="_")
}

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

Now we can use the 'get_variable_measures_from_acmt' function (see code to create this function above) to pull the variables.

```
##Pull relevant ACS variables
address_to_radius_to_variable_to_measures_list <-get_variable_measures_from_acmt(address_to_lat_long_li
address_ndi_measures<-address_to_radius_to_variable_to_measures_list
```

Next, we tranform the list of measures back to a dataset

```
#reshape list of measures to dataset
##Convert to dataframe

convert_to_dataframe <- function (address_ndi_measures) {
  address_vector <- c()
  radius_vector <- c()
  variable_vector <- c()
  value_vector <- c()

  for(address in names(address_ndi_measures)){
    for(radius_character in names(address_ndi_measures[[address]])){
      for(variable in names(address_ndi_measures[[address]][[radius_character]])){
        address_vector <- c(address_vector, address)
        radius_vector <- c(radius_vector, as.numeric(radius_character))
        variable_vector <- c(variable_vector, variable)
        value_vector <- c(value_vector, address_ndi_measures[[address]][[radius_character]][[variable]]
      }
    }
  }

  dataframe_address <- data.frame(address_vector, radius_vector, variable_vector, value_vector, strings
  names(dataframe_address) <- c("address", "radius", "variable", "value")
  return(dataframe_address)
}

dataframe_address <- convert_to_dataframe(address_ndi_measures = address_to_radius_to_variable_to_measu

dataframe_address$radius<-NULL
address_wide_dataframe<-reshape(dataframe_address, v.names="value",timevar = "variable", idvar = "addre
```

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

```
adi_measure_dataset <- address_wide_dataframe %>%
  mutate(less_than_9yrs_education= ((value.no_education_count +
```

```
                                    value.pre_school_count + value.kindergarten_count +
                                    value.first_grade_count + value.second_grade_count +
                                    value.third_grade_count + value.fourth_grade_count +
                                    value.fifth_grade_count + value.sixth_grade_count +
                                    value.seventh_grade_count + value.eighth_grade_count +
                                    value.ninth_grade_count) / value.pop_25_and_over_count),
        hs_education_or_more = (value.high_school_grad_count +
                                  value.ged_or_alt_diploma_count + value.some_college_less_than_1_yea
                                  value.some_college_1_year_or_more_count + value.associates_degree_c
                                  value.bachelors_degree_count + value.masters_degree_count +
                                  value.professional_degree_count + value.doctoral_degree_count)/
           value.pop_25_and_over_count,
        white_collar_occ=(value.males_in_professional_occup_count +
                            value.males_in_management_count +
                            value.females_in_professional_occup_count +
                            value.females_in_management_count) / value.males_16_older_workforce_count,
        income_disparity = (100*(value.hhincome_less_than_10k_count/
                                  (value.hhincome_50k_to_59k_count + value.hhincome_60k_to_74k_count
                                     value.hhincome_75k_to_99k_count + value.hhincome_100k_to_124k_co
                                     value.hhincome_125k_to_149k_count + value.hhincome_150k_to_199k_
                                     value.hhincome_200k_or_more_count))),
                                 below_150_poverty = (value.pop_below_100_poverty_threshold_count +
                                                        value.pop_100_to_149_poverty_threshold_count)/
                                 single_parent_with_kids = (value.female_head_kids_count +
                                                              value.male_householder_kids_count)/value
                                 hh_novehicle = (value.owner_no_vehicle_count +
                                                   value.renter_no_vehicle_count)/ value.occupied_hous
                                 hh_no_phone = (value.no_phone_owner_count + value.no_phone_renter_cou
                                   value.total_occupied_housing_units_tele_count,
                                 hh_no_plumb = (value.no_comp_plumb_owner_count+value.no_comp_plumb_re
                                   value.total_occupied_housing_units_plumb_count,
                                 hh_crowded=(value.owner_1.01_to_1.5_per_room_count +
                                               value.owner_1.51_to_2.0_per_room_count +
                                               value.owner_2.01_or_more_per_room_count +
                                               value.renter_1.01_to_1.5_per_room_count +
                                               value.renter_1.51_to_2.0_per_room_count +
                                               value.renter_2.01_or_more_per_room_count)/
                                   value.total_occupied_housing_units_room_count)
```

Next we can run a principal component analysis to find the weights to calculate the measure.

```
adi_measures<-adi_measure_dataset %>%
  subset(select=c(address, less_than_9yrs_education, hs_education_or_more, value.pop_25_and_over_count,
                  white_collar_occ, income_disparity, below_150_poverty, single_parent_with_kids,
                  hh_novehicle, hh_no_phone, hh_no_plumb, hh_crowded, value.med_hincome_count, value.me

adi_pca<-princomp(adi_measures[2:18], cor=TRUE)
summary(adi_pca)
```

```
## Importance of components:
##                            Comp.1    Comp.2     Comp.3     Comp.4     Comp.5
## Standard deviation     2.7587962 2.0569866 1.18207990 1.07508276 0.98287804
## Proportion of Variance 0.4477033 0.2488938 0.08219488 0.06798841 0.05682643
## Cumulative Proportion  0.4477033 0.6965971 0.77879196 0.84678037 0.90360680
```

```
##                             Comp.6     Comp.7     Comp.8     Comp.9     Comp.10
## Standard deviation     0.81197985 0.55481396 0.49638804 0.45303434 0.301772822
## Proportion of Variance 0.03878302 0.01810697 0.01449418 0.01207295 0.005356873
## Cumulative Proportion  0.94238981 0.96049679 0.97499097 0.98706392 0.992420789
##                             Comp.11     Comp.12    Comp.13     Comp.14
## Standard deviation     0.225561609 0.189887158 0.15232520 0.120669922
## Proportion of Variance 0.002992826 0.002121008 0.00136488 0.000856543
## Cumulative Proportion  0.995413615 0.997534623 0.99889950 0.999756046
##                              Comp.15      Comp.16      Comp.17
## Standard deviation     0.0484192772 0.0423889367 2.442025e-03
## Proportion of Variance 0.0001379074 0.0001056954 3.507933e-07
## Cumulative Proportion  0.9998939538 0.9999996492 1.000000e+00
```

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

```
##              less_than_9yrs_education                  hs_education_or_more
##                            0.30976330                           -0.31990245
##           value.pop_25_and_over_count                      white_collar_occ
##                           -0.20892915                           -0.33137692
##                      income_disparity                     below_150_poverty
##                            0.21572481                            0.07049906
##               single_parent_with_kids                         hh_novehicle
##                            0.31462710                           -0.10322106
##                           hh_no_phone                           hh_no_plumb
##                            0.11812442                            0.06870657
##                            hh_crowded               value.med_hincome_count
##                            0.22652401                           -0.31232942
##               value.median_rent_count           value.median_mortgage_count
##                           -0.26639097                           -0.31596430
##              value.med_home_val_count           value.unemployed_proportion
##                           -0.31643281                            0.14835464
## value.owner_occupied_units_proportion
##                            0.16316184
```

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.

```
##assign loading values for each variable
less_than_9yrs_loading<-adi_pca$loadings[1]
hs_education_or_more_loading<-adi_pca$loadings[2]
value.pop_25_and_over_count_loading<-adi_pca$loadings[3]
white_collar_occ_loading<-adi_pca$loadings[4]
income_disparity_loading<-adi_pca$loadings[5]
below_150_poverty_loading<-adi_pca$loadings[6]
single_parent_with_kids_loading<-adi_pca$loadings[7]
hh_novehicle_loading<-adi_pca$loadings[8]
hh_no_phone_loading<-adi_pca$loadings[9]
hh_no_plumb_loading<-adi_pca$loadings[10]
hh_crowded_loading<-adi_pca$loadings[11]
value.med_hincome_count_loading<-adi_pca$loadings[12]
value.median_rent_count_loading<-adi_pca$loadings[13]
value.median_mortgage_count_loading<-adi_pca$loadings[14]
value.med_home_val_count_loading<-adi_pca$loadings[15]
value.unemployed_proportion_loading<-adi_pca$loadings[16]
value.owner_occupied_units_proportion_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)+
           (value.pop_25_and_over_count_loading*value.pop_25_and_over_count)+
           (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)+
           (value.med_hincome_count_loading*value.med_hincome_count)+
           (value.med_home_val_count_loading*value.med_home_val_count)+
           (value.unemployed_proportion_loading*value.unemployed_proportion)+
           (value.owner_occupied_units_proportion_loading*value.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)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -1.4015 -0.9080 -0.1725  0.0000  0.7605  1.6440
```

```
adi_mean_table<-adi_measures %>%
  group_by(address) %>%
  summarise_at(vars(adi_standardized), list(adi_mean=mean))

adi_mean_table
```

```
## # A tibble: 17 x 2
##    address                                adi_mean
##    <chr>                                     <dbl>
##  1 Alan T. Sugiyama High School              1.22
##  2 Ballard High School                      -0.928
##  3 Chief Sealth International High School    0.998
##  4 Cleveland High School                     0.761
##  5 Franklin High School                      1.64
##  6 Garfield High School                     -0.735
##  7 Ingraham High School                      0.580
##  8 Interagency Academy                       0.351
##  9 Lincoln High School                      -1.40
## 10 Nathan Hale High School                  -0.173
## 11 Nova                                     -0.924
## 12 Ranier Beach High School                  1.38
## 13 Roosevelt High School                    -1.40
## 14 Seattle World School                     -0.458
## 15 Skills Center                             0.554
## 16 The Center School                        -0.566
## 17 West Seattle High School                 -0.908
```

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', 'B25003_002', 'B25002_003', 'B25002_001','B11012_001', 'B11012_008', 'B1

#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
acs_count_names<-paste(acsvars$var_name, "count", sep="_")
if (length(acsvars$var_name[acsvars$universe_col != ""]) == 0) {   # prevent having something that is e
  acs_proportion_names <- character(0)
} else {
  acs_proportion_names <- paste(acsvars$var_name[!is.na(acsvars$universe_col)], "proportion", sep="_")
}
```

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.

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

##Pull relevant ACS variables
address_to_radius_to_variable_to_measures_list <-get_variable_measures_from_acmt(address_to_lat_long_li
address_adi_measures<-address_to_radius_to_variable_to_measures_list
```

Once the measures are pulled for each location, we can transform the list into a wide dataframe for easier analysis.

```
#reshape list of measures to dataset
##Convert to dataframe

convert_to_dataframe <- function (address_adi_measures) {
  address_vector <- c()
  radius_vector <- c()
  variable_vector <- c()
  value_vector <- c()

  for(address in names(address_adi_measures)){
    for(radius_character in names(address_adi_measures[[address]])){
      for(variable in names(address_adi_measures[[address]][[radius_character]])){
        address_vector <- c(address_vector, address)
        radius_vector <- c(radius_vector, as.numeric(radius_character))
```

```
        variable_vector <- c(variable_vector, variable)
        value_vector <- c(value_vector, address_adi_measures[[address]][[radius_character]][[variable]]
      }
    }
  }

  dataframe_address <- data.frame(address_vector, radius_vector, variable_vector, value_vector, strings
  names(dataframe_address) <- c("address", "radius", "variable", "value")
  return(dataframe_address)
}

dataframe_address <- convert_to_dataframe(address_adi_measures = address_to_radius_to_variable_to_measu

dataframe_address$radius<-NULL
address_wide_dataframe<-reshape(dataframe_address, v.names="value",timevar = "variable", idvar = "addres
```

Once the dataset is transformed, we can calculate the measures using the ACS variables that were pulled.

```
sfi_measures<-address_wide_dataframe %>%
  mutate(housing_not_owner_occupied = (value.occupied_units_count - value.owner_occupied_units_count)/va
         living_alone = (value.living_alone_female_proportion + value.living_alone_male_proportion)/valu
         recent_move = (value.renter_occupied_recent_move_count+value.owner_occupied_recent_move_count),
  mutate(housing_not_owner_occupied_stand = (housing_not_owner_occupied-mean(housing_not_owner_occupied
         vacant_housing_stand = (value.vacant_units_proportion-mean(value.vacant_units_proportion))/sd(v
         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_alone_sta

summary(sfi_measures$sfi_value)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -3.1495 -1.1655  0.1305  0.0000  1.3953  3.1756
```

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.
## -3.1495 -1.1655  0.1305  0.0000  1.3953  3.1756
```

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

sfi_mean_table
```

```
## # A tibble: 17 x 2
##    address                                 sfi_mean
##    <chr>                                      <dbl>
##  1 Alan T. Sugiyama High School              -0.129
##  2 Ballard High School                       -1.17
##  3 Chief Sealth International High School    -1.22
##  4 Cleveland High School                     -2.18
##  5 Franklin High School                       1.68
##  6 Garfield High School                       0.537
```

```
##  7 Ingraham High School                    -0.954
##  8 Interagency Academy                     -0.991
##  9 Lincoln High School                      1.40
## 10 Nathan Hale High School                 -2.88
## 11 Nova                                     0.186
## 12 Ranier Beach High School                 0.355
## 13 Roosevelt High School                    0.130
## 14 Seattle World School                     2.40
## 15 Skills Center                            2.82
## 16 The Center School                        3.18
## 17 West Seattle High School                -3.15
```

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', 'B25001_001', 'B11001_001', 'B17002_002', 'B17002_001', 'B23025_001', 'B

#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
acs_count_names<-paste(acsvars$var_name, "count", sep="_")
if (length(acsvars$var_name[acsvars$universe_col != ""]) == 0) {   # prevent having something that is e
  acs_proportion_names <- character(0)
} else {
  acs_proportion_names <- paste(acsvars$var_name[!is.na(acsvars$universe_col)], "proportion", sep="_")
}
```

Once the variables are designated, we can pull the relevant ACS values for each location.

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

##Pull relevant ACS variables
address_to_radius_to_variable_to_measures_list <-get_variable_measures_from_acmt(address_to_lat_long_li
address_svi_measures<-address_to_radius_to_variable_to_measures_list
```

Next we can transform the list of values into a wide dataset.

```
#reshape list of measures to dataset
##Convert to dataframe

convert_to_dataframe <- function (address_svi_measures) {
  address_vector <- c()
  radius_vector <- c()
  variable_vector <- c()
  value_vector <- c()

  for(address in names(address_svi_measures)){
    for(radius_character in names(address_svi_measures[[address]])){
      for(variable in names(address_svi_measures[[address]][[radius_character]])){
        address_vector <- c(address_vector, address)
        radius_vector <- c(radius_vector, as.numeric(radius_character))
        variable_vector <- c(variable_vector, variable)
        value_vector <- c(value_vector, address_svi_measures[[address]][[radius_character]][[variable]]
      }
    }
  }

  dataframe_address <- data.frame(address_vector, radius_vector, variable_vector, value_vector, strings
  names(dataframe_address) <- c("address", "radius", "variable", "value")
  return(dataframe_address)
}

dataframe_address <- convert_to_dataframe(address_svi_measures = address_to_radius_to_variable_to_measu

dataframe_address$radius<-NULL
address_wide_dataframe<-reshape(dataframe_address, v.names="value",timevar = "variable", idvar = "addres
```

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(address_wide_dataframe)-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
address_wide_dataframe$value.unemployed_proportion
svi_measures<-address_wide_dataframe %>%
  #First create the estimates
  mutate(e_totpop=value.total_pop_count,
         e_hu=value.housing_units_count,
         e_hh=value.total_households_count,
         e_pov=value.below_pov_count,
         e_unemp=value.unemployed_count,
         e_pci=value.med_hincome_count,
         e_nohsdp=value.no_hsdiploma_count,
         e_age65=(value.males_65_to_74_count + value.males_75_to_84_count+ value.males_85_and_older_cou
         e_age17=(value.males_under_5_count+value.males_5_to_9_count+value.males_10_to_14_count+value.ma
```

```
        e_disabl=(value.males_under5_disability_count+value.males_5_17_disability_count+value.males_18
        e_sngpnt=(value.male_householder_kids_count+value.female_head_kids_count),
        e_minrty=(value.total_pop_count-value.non_hisp_white_count),
        e_limeng=(value.ntv_sp_lmt_eng_notwell_count+value.ntv_sp_lmt_eng_notatall_count+value.ntv_ie_l
                              value.ntv_ie_lmt_eng_notatall_count+value.ntv_api_lmt_en_notwell_count+val
                              value.ntv_oth_lmt_en_notwell_count+value.ntv_oth_lmt_en_notatall_count+val
                              value.fb_sp_lmt_eng_notatall_count+value.fb_ie_lmt_eng_notwell_count+value
                              value.fb_api_lmt_en_notwell_count+value.fb_api_lmt_en_notatall_count+value
                              value.fb_oth_lmt_en_notatall_count),
        e_munit=(value.units_10_to_19_count+value.units_20_to_49_count+value.units_50_ormore_count)/val
        e_mobile=value.mobile_homes_count,
         e_crowd=(value.owner_1.01_to_1.5_per_room_count+value.owner_1.51_to_2.0_per_room_count+value.
        e_noveh=(value.owner_no_vehicle_count+value.renter_no_vehicle_count)/value.occupied_housing_veh
        e_groupq=value.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=value.below_pov_proportion,
        ep_unemp=value.unemployed_proportion*100,
        ep_pci=value.med_hincome_count*100,
        ep_nohsdp=value.no_hsdiploma_proportion*100,
        ep_age65=(e_age65/e_totpop)*100,
        ep_age17=(e_age17/e_totpop)*100,
        ep_disabl=(e_disabl/value.civilian_pop_count)*100,
        ep_sngpnt=(e_sngpnt/value.total_households_count)*100,
        ep_minrty=(e_minrty/e_totpop)*100,
        ep_limeng=(e_limeng/value.total_ages_5_up_count)*100,
        ep_munit=(e_munit/e_hu)*100,
        ep_mobile=value.mobile_homes_proportion*100,
        ep_crowd=(e_crowd/value.total_occupied_housing_units_room_count)*100,
        ep_noveh=(e_noveh/value.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 g
    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   2.000   1.765   3.000   6.000
```

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

svi_mean_table
```

```
## # A tibble: 17 x 2
##    address                                svi_mean
##    <chr>                                     <dbl>
##  1 Alan T. Sugiyama High School                  3
##  2 Ballard High School                           0
##  3 Chief Sealth International High School        0
##  4 Cleveland High School                         3
##  5 Franklin High School                          4
##  6 Garfield High School                          0
##  7 Ingraham High School                          2
##  8 Interagency Academy                           2
##  9 Lincoln High School                           0
## 10 Nathan Hale High School                       0
## 11 Nova                                          1
## 12 Ranier Beach High School                      6
## 13 Roosevelt High School                         2
## 14 Seattle World School                          1
## 15 Skills Center                                 4
## 16 The Center School                             0
## 17 West Seattle High School                      2
```

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).