Methods Supplement

Julie Wisch

December 2022

Contents

Package Installation

```
## If a package is installed, it will be loaded. If any
## are not, the missing package(s) will be installed
## from CRAN and then loaded.

## Packages required for this script
packages = c("tidycensus", "data.table", "ggplot2", "gridExtra",
             "sf", "maptools", "spatstat", "spdep", "RColorBrewer",
             "ecespa", "censusr", "raster", "concaveman", "ggpubr")

## Now load or install&load all
```

```
package.check <- lapply(
  packages,
  FUN = function(x) {
    if (!require(x, character.only = TRUE)) {
      install.packages(x, dependencies = TRUE)
      library(x, character.only = TRUE)
    }
  }
)

rm(package.check, packages)
```

Area Deprivation Index

The University of Wisconsin offers neighborhood level estimates of area deprivation (https://www.neighborhoodatlas.medicine.wisc.edu/). These tract-level values are freely available for download. Here we demonstrate an R-based way to replicate the Arc-GIS based mappings that are displayed on the University of Wisconsin's website.

We have pre-downloaded ADI for the state of Missouri and include it in the git repo. Other state values can be downloaded directly from the Neighborhood Atlas website.

```
ADI <- read.csv(".././Data/MO_2019_ADI_CensusBlockGroup_v3.1.csv")
```

In order to generate a plot that displays ADI at the tract level for the city of St. Louis, we need a shape file for the city. There are a myriad of ways to acquire this, but we will use the ACS geometries that we will rely on later for population estimation.

In order to use tidycensus, you must acquire your own census API key. This can be obtained from https://api.census.gov/data/key_signup.html

After establishing your relationship with the census API, you can pull the geometries using the following call.

```
metro_region <- get_acs(geography = "block group"
                        , state = c("MO")
                        ,
                        county = c("St. Louis city")
                        ,
                        variables = "B19013_001"
                        ,
                        geometry = TRUE)
```

Then we merge the ADI data to the appropriate census tract to facilitate plotting

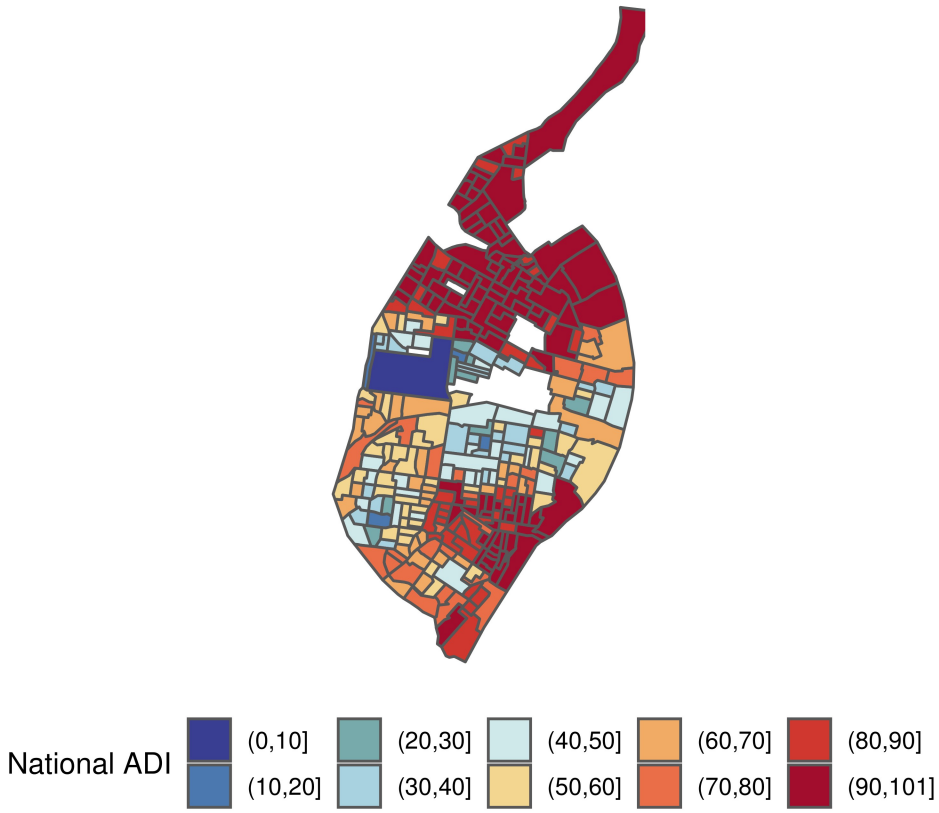
```
ADI$ADI_NATRANK <- as.numeric(as.character(ADI$ADI_NATRANK))
ADI <- ADI[, c("GISJOIN", "ADI_NATRANK")]

# #How to convert between available values https://www.nhgis.org/geographic-crosswalks
ADI$GEOID <- paste0(substr(ADI$GISJOIN, start = 2, stop = 3),
                    substr(ADI$GISJOIN, start = 5, stop = 7),#state & county FIPScode
                    substr(ADI$GISJOIN,
```

```
                          start = nchar(ADI$GISJOIN) -6,
                          stop = nchar(ADI$GISJOIN))) #census tract code
                                                      #followed by census
                                                      #block code

#Generating a dataframe for plotting
plot_df <- merge(metro_region, ADI[, c("GEOID", "ADI_NATRANK")], by = "GEOID")
#Classifying ADI by National Deciles
plot_df$ADI_cut <- as.factor(cut(plot_df$ADI_NATRANK,
                                 c(0, 10, 20, 30, 40, 50, 60, 70, 80, 90, 101)))
```

Generating Figure 1A

Then we can plot the ADI distribution. This helps us understand the structure of St. Louis. Affluence is concentrated along the central corridor, exclusive of a highly industrialized region that appears in white as ADI is not available for that tract. Areas of greatest deprivation exist in the northern portion of the city, as well as the southeastern portion.

```
ggplot(plot_df[!is.na(plot_df$ADI_cut),]) + geom_sf(aes(fill = ADI_cut)) +
  #Manually selecting hex colors that match those used on UW's site
  scale_fill_manual(values = c("#3a3f94", "#4c78b0",
                               "#77aaab",
                               "#a8d2e0", "#cfe9ea", "#f3d890",
                               "#f2ab65", "#ea704a", "#d03830",
                               "#a30d2f"), name = "National ADI") +
  xlim(c(-90.33, -90.18)) + ylim(c(38.53, 38.785)) + theme_void() +
  theme(legend.position = "bottom")
```



Sample and Population Comparisons

Loading Local Data

Note that due to data sharing restrictions, we do not provide actual data in this example. Instead, we have generated simulated data for the purposes of users interested in running this example. Because this is simulated data, the results differ somewhat from the actual results presented in the manuscript

```
#reading in prepped data
participants <- readRDS(".././Data/Synthetic_Participant_Data.RDS") #synthetic data
colnames(participants)[8:9] <- c( "lon", "lat")
participants <- data.frame(participants)
```

Population Estimate Extraction from ACS

Using Tidycensus, you can download the list of all variables provided by the ACS using the call below.

```
#Loading a list of all variables for the chosen year
vars <- load_variables(year = 2020,
                       dataset = "acs5",
                       cache = TRUE)
```

Because we are working with an adult population, we want to limit the city population we evaluate to only those aged 21 - 84 (based on the available ACS age based stratifications, this is the closest we can get to our sample age range). We will limit the list of ACS variables we are looking at to those that include an age-based stratification, and then we will view the list of variables available so we can figure out how to extract population count, age, sex, education and race.

```
vars <- dplyr::filter(vars, grepl('AGE', concept))
View(vars)
```

For the 2019 ACS data, the following are the relevant demographic variables we elected to extract:

- B01001_002 - B01001_007 - male population by age by tract
- B01001_026 - B01001_031 - female population by age by tract
- B01001A_002 - B01001A_007 - white only male population by age by tract
- B01001A_018 - B01001A_022 - white only female population by age by tract
- B01001B_002 - B01001B_007 - black only male population by age by tract
- B01001B_018 - B01001B_022 - black only female population by age by tract
- B16010_002 – total less than high school graduate
- B16010_015 – high school graduate
- B16010_028 – some college
- B16010_041 – BS or higher

```
stl_value_ed <- get_acs(geography = "tract"
                        , state = c("MO")
                        ,
                        county = c("St. Louis city")
                        ,
                        variables = c("B16010_002", #less than hs
                                      "B16010_015", #hs
                                      "B16010_028", #some college
                                      "B16010_041") #bs or more
                        ,
                        geometry = TRUE,
                        year = 2019)

stl_amended_values <- get_acs(geography = "block group"
                        , state = c("MO")
                        ,
                        county = c("St. Louis city")
                        ,
                        variables = c("B01001_002", "B01001_003",
                                      "B01001_004", #males by age
                                      "B01001_005", "B01001_006",
                                      "B01001_007", #males by age
                                      "B01001_026", "B01001_027",
                                      "B01001_028", #females by age
                                      "B01001_029", "B01001_030", "B01001_031")
                        ,
                        geometry = TRUE,
                        year = 2019)


stl_amended_values_race <- get_acs(geography = "tract"
```

```
                      , state = c("MO")
                      ,
                      county = c("St. Louis city")
                      ,
                      variables = c(
                                    "B01001A_002", "B01001A_017",
                                    "B01001A_003", "B01001A_004",
                                    "B01001A_005", "B01001A_006",
                                    "B01001A_007", "B01001A_018",
                                    "B01001A_019", "B01001A_020",
                                    "B01001A_021", "B01001A_022",
                                    "B01001B_002", "B01001B_017",
                                    "B01001B_003", "B01001B_004",
                                    "B01001B_005", "B01001B_006",
                                    "B01001B_007", "B01001B_018",
                                    "B01001B_019", "B01001B_020",
                                    "B01001B_021", "B01001B_022")
                      ,
                      geometry = TRUE,
                      year = 2019)

stl_amended_values_age <- get_acs(geography = "block group"
                      , state = c("MO")
                      ,
                      county = c("St. Louis city")
                      ,
                      variables = c("B01001_008", "B01001_009", "B01001_010",
                                    "B01001_011", "B01001_012", "B01001_013",
                                    "B01001_014", "B01001_015", "B01001_016",
                                    "B01001_017", "B01001_018", "B01001_019",
                                    "B01001_020", "B01001_021", "B01001_022",
                                    "B01001_023", "B01001_024",
                                    "B01001_032", "B01001_033", "B01001_034",
                                    "B01001_035", "B01001_036", "B01001_037",
                                    "B01001_038", "B01001_039", "B01001_040",
                                    "B01001_041", "B01001_042", "B01001_043",
                                    "B01001_044", "B01001_045", "B01001_046",
                                    "B01001_047", "B01001_048")
                      ,
                      geometry = TRUE,
                      year = 2019)
```

We have to then aggregate this extracted data to get the total counts by sex of individuals aged 21 - 84.

```
total_num_males_under_20 <- data.table(stl_amended_values[stl_amended_values$variable %in%
                                          c("B01001_003", "B01001_004", #males by age
                                           "B01001_005", "B01001_006", "B01001_007"),])[, sum(estimate

total_num_females_under_20 <- data.table(stl_amended_values[stl_amended_values$variable %in%
                                            c("B01001_027", "B01001_028", #females by age
                                            "B01001_029", "B01001_030", "B01001_031"),])[, sum(estima

total_counts_by_sex <- merge(data.frame(stl_amended_values)[stl_amended_values$variable ==
```

```
                                                        "B01001_002",
                                                      c("GEOID", "estimate")],
                                                       total_num_males_under_20,
                                                      by = "GEOID")

total_counts_by_sex <- merge(total_counts_by_sex, data.frame(stl_amended_values)[stl_amended_values$var
                                                        "B01001_026",
                                                        c("GEOID", "estimate")],
                                                        by = "GEOID")

total_counts_by_sex <- merge(total_counts_by_sex, total_num_females_under_20,
                             by = "GEOID")

colnames(total_counts_by_sex) <- c("GEOID", "males_total", "males_under_20",
                                   "females_total", "females_under_20")

total_counts_by_sex$total_above_19 <- total_counts_by_sex$males_total +
                                      total_counts_by_sex$females_total -
                                      total_counts_by_sex$males_under_20 -
                                      total_counts_by_sex$females_under_20

stl_amended_values <- merge(stl_amended_values,
                            total_counts_by_sex[, c("GEOID", "total_above_19")],
                            by = "GEOID")
```

Similarly, we aggregated extracted data to get the total counts by race of individuals aged 21 - 84.

```
total_num_whites_under_20 <- data.table(stl_amended_values_race
                                        [stl_amended_values_race$variable %in%
                                            c("B01001A_003", "B01001A_004",
                                            "B01001A_005", "B01001A_006",
                                            "B01001A_007",
                                            "B01001A_018", "B01001A_019",
                                            "B01001A_020", "B01001A_021",
                                            "B01001A_022"),])[, sum(estimate,
                                                                    na.rm = TRUE),
                                                              by = GEOID]
total_num_blacks_under_20 <- data.table(stl_amended_values_race
                                        [stl_amended_values_race$variable %in%
                                            c("B01001B_003", "B01001B_004",
                                              "B01001B_005", "B01001B_006",
                                              "B01001B_007",
                                              "B01001B_018", "B01001B_019",
                                              "B01001B_020", "B01001B_021",
                                              "B01001B_022"),])[, sum(estimate,
                                                                      na.rm = TRUE),
                                                                by = GEOID]

total_num_whites <- data.table(stl_amended_values_race
                               [stl_amended_values_race$variable %in%
                                   c("B01001A_002", "B01001A_017"),])[, sum(estimate,
                                                                            na.rm = TRUE),
                                                                      by = GEOID]
```

```
total_num_blacks <- data.table(stl_amended_values_race
                               [stl_amended_values_race$variable %in%
                                   c("B01001B_002", "B01001B_017"),])[, sum(estimate,
                                                                            na.rm = TRUE),
                                                                      by = GEOID]

total_counts_by_race <- merge(total_num_whites, total_num_blacks,
                              by = "GEOID")
total_counts_by_race <- merge(total_counts_by_race, total_num_whites_under_20,
                              by = "GEOID")
total_counts_by_race <- merge(total_counts_by_race, total_num_blacks_under_20,
                              by = "GEOID")
colnames(total_counts_by_race) <- c("GEOID", "white_total", "black_total",
                                    "white_under_20", "black_under_20")


total_counts_by_race$white_above_19 <- total_counts_by_race$white_total -
                                           total_counts_by_race$white_under_20
total_counts_by_race$black_above_19 <- total_counts_by_race$black_total -
                                           total_counts_by_race$black_under_20
```

And aggregate by education level

```
stl_value_ed <- data.table(stl_value_ed)[,sum(estimate), by = variable]
```

It's not actually possible to know the median age, exclusive of individuals aged 0 - 20, 85+ with the existing structure of the ACS data. Instead, we assume that the median age of each 5 year age bin is exactly in the middle, and calculate the estimated median age by tract using extracted data. Given the relative smallness of the age bands for the ACS (5 years), as well as the high spatial resolution (tract level), we assume that this calculation will be within the margin of error for ACS in urban environments.

```
stl_amended_values_age <- stl_amended_values_age[, !(names(stl_amended_values_age) %in% "moe")]
stl_amended_values_age <- data.frame(stl_amended_values_age)
stl_amended_values_age <- tidyr::spread(stl_amended_values_age, variable, estimate)

stl_amended_values_age$mean_age <- (stl_amended_values_age$B01001_008*20 +
                                      stl_amended_values_age$B01001_009 * 21 +
                                      stl_amended_values_age$B01001_010 * 23 +
                                      stl_amended_values_age$B01001_011 * 27 +
                                      stl_amended_values_age$B01001_012 * 32 +
                                      stl_amended_values_age$B01001_013 * 37 +
                                      stl_amended_values_age$B01001_014 * 42 +
                                      stl_amended_values_age$B01001_015 * 47 +
                                      stl_amended_values_age$B01001_016 * 52 +
                                      stl_amended_values_age$B01001_017 * 57 +
                                      stl_amended_values_age$B01001_018 * 62 +
                                      stl_amended_values_age$B01001_019 * 67 +
                                      stl_amended_values_age$B01001_020 * 72 +
                                      stl_amended_values_age$B01001_021 * 77 +
                                      stl_amended_values_age$B01001_023 * 82 +
                                      stl_amended_values_age$B01001_032 * 20 +
                                      stl_amended_values_age$B01001_033 * 21 +
                                      stl_amended_values_age$B01001_034 * 23 +
```

```
                                    stl_amended_values_age$B01001_035 * 27 +
                                    stl_amended_values_age$B01001_036 * 32 +
                                    stl_amended_values_age$B01001_037 * 37 +
                                    stl_amended_values_age$B01001_038 * 42 +
                                    stl_amended_values_age$B01001_039 * 47 +
                                    stl_amended_values_age$B01001_040 * 52 +
                                    stl_amended_values_age$B01001_041 * 57 +
                                    stl_amended_values_age$B01001_042 * 62 +
                                    stl_amended_values_age$B01001_043 * 67 +
                                    stl_amended_values_age$B01001_044 * 72 +
                                    stl_amended_values_age$B01001_045 * 77 +
                                    stl_amended_values_age$B01001_046 * 82) /
                                    rowSums(stl_amended_values_age[, 4 : 37])

summed_df <- colSums(stl_amended_values_age[, 4:37])
summed_vec <- c(rep(20, summed_df[1] + summed_df[16]),
                rep(21, summed_df[2]+ summed_df[17]),
                rep(23, summed_df[3]+ summed_df[18]),
                rep(27, summed_df[4]+ summed_df[19]),
                rep(32, summed_df[5]+ summed_df[20]),
                rep(37, summed_df[6]+ summed_df[21]),
                rep(42, summed_df[7]+ summed_df[22]),
                rep(47, summed_df[8]+ summed_df[23]),
                rep(52, summed_df[9]+ summed_df[24]),
                rep(57, summed_df[10]+ summed_df[25]),
                rep(62, summed_df[11]+ summed_df[26]),
                rep(67, summed_df[12]+ summed_df[27]),
                rep(72, summed_df[13]+ summed_df[28]),
                rep(77, summed_df[14]+ summed_df[29]),
                rep(82, summed_df[15]+ summed_df[30]))
```

Now we compare the population data with our participants in order to make a demographic comparison.

```
#Age
print(paste0("Population Mean Age = ", round(mean(summed_vec), 2),
             " Population Age SD = ", round(sd(summed_vec), 2),
             " Sample Mean Age = ", round(mean(participants$TrueAge)),
             " Sample Age SD = ", round(sd(participants$TrueAge), 2)))
```

```
## [1] "Population Mean Age = 47.19 Population Age SD = 16.61 Sample Mean Age = 53 Sample Age SD = 11.4
```

```
#Perform a t test to compare population and sample

print(paste0("T Test for Age-related differences: ",
             t.test2(mean(summed_vec), mean(participants$TrueAge),
                         sd(summed_vec), sd(participants$TrueAge),
                         length(summed_vec), length(participants$TrueAge))))
```

```
## [1] "T Test for Age-related differences: -6.00371560520809"
## [2] "T Test for Age-related differences: 0.743078858635265"
## [3] "T Test for Age-related differences: -8.07951341292967"
## [4] "T Test for Age-related differences: 0.000000000000032156135775322 9"
```

```r
#Chi Square test for differences by Sex
sex_chi <- data.frame("STL_City" = c(sum(total_counts_by_sex$males_total -
                                           total_counts_by_sex$males_under_20),
                                     sum(total_counts_by_sex$females_total -
                                           total_counts_by_sex$females_under_20)),
                      "Cohort" = c(length(participants[participants$Sex ==
                                                         "male", "Sex"]),
                                   length(participants[participants$Sex ==
                                                         "female", "Sex"])))
print("Test for differences by sex: ")
```

```
## [1] "Test for differences by sex: "
```

```r
chisq.test(sex_chi)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  sex_chi
## X-squared = 23.578, df = 1, p-value = 0.000001199
```

```r
#Chi Square test for differences by race
race_chi <- data.frame("STL_City" = c(sum(total_counts_by_race$white_above_19),
                                      sum(total_counts_by_race$black_above_19)),
                       "Cohort" = c(length(participants[participants$Race ==
                                                          "White", "Race"]),
                                    length(participants[participants$Race ==
                                                          "Black", "Race"])))
print("Test for differences by race: ")
```

```
## [1] "Test for differences by race: "
```

```r
chisq.test(race_chi)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  race_chi
## X-squared = 21.715, df = 1, p-value = 0.000003163
```

```r
#Chi Square test for differences by education
ed_chi <- data.frame("STL_City" = stl_value_ed$V1,
                       "Cohort" = as.vector(table(cut(participants$Education,
                                                      breaks = c(0, 11.5, 12.5,
                                                                 15.5, 100)))))
print("Test for differences by education: ")
```

```
## [1] "Test for differences by education: "
```

```
chisq.test(ed_chi)
```

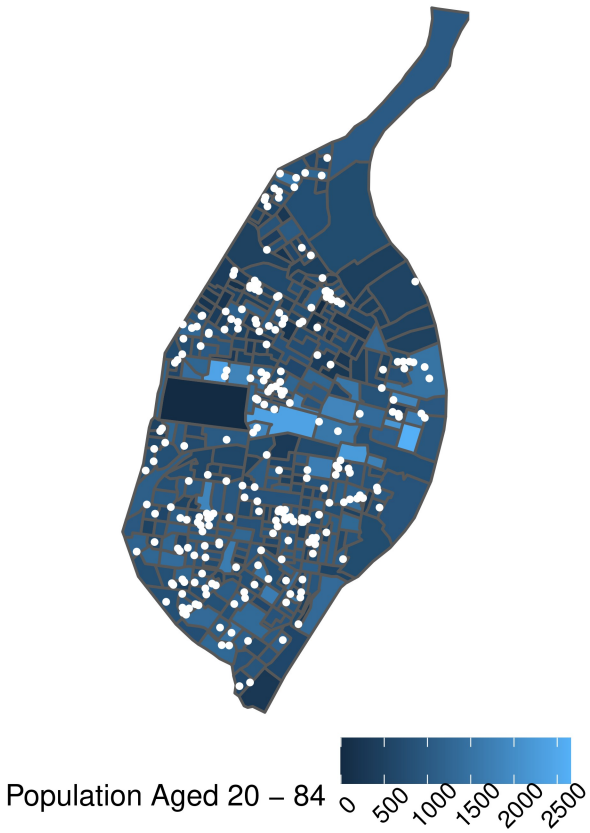
```
##
##  Pearson's Chi-squared test
##
## data:  ed_chi
## X-squared = 4.5762, df = 3, p-value = 0.2056
```

```
rm(stl_amended_values_age, stl_amended_values_race, stl_value_ed,
   total_counts_by_race, total_counts_by_sex, total_num_blacks,
   total_num_blacks_under_20,
   total_num_females_under_20, total_num_males_under_20, total_num_whites,
   total_num_whites_under_20,
   vars)

rm(ed_chi, race_chi, sex_chi)
```

Generating Figure 1B

We present the sampled distribution overlain on tract level population information.

```
#Population count by tract
stl_value <- get_acs(geography = "block group"
                     , state = c("MO")
                     ,
                     county = c("St. Louis city")
                     ,
                     variables = c("B01001_001")                    ,
                     geometry = TRUE,
                     year = 2019)

#getting participant count by census tract
participants_count <- data.frame(data.table(participants)
                                 [, .N, by = GEOID])
participants_count$GEOID <- substr(participants_count$GEOID,
                                   start = 1, stop = 12)

participants_poly <- merge(participants_count,
                           stl_amended_values[, c("GEOID", "NAME", "geometry")],
                           by = "GEOID", all = TRUE)
participants_poly <- participants_poly[!duplicated(participants_poly),]
participants_poly[is.na(participants_poly)] <- 0
participants_poly <- st_as_sf(participants_poly)

ggplot(stl_value) +
  aes(fill = estimate) + geom_sf() +
  xlim(c(-90.32, -90.175)) + ylim(c(38.54, 38.77)) + theme_void() +
  theme(legend.position = "bottom", legend.text = element_text(angle = 45))  +
  geom_point(data = participants, aes(x = lon, y = lat, fill = 1),
             colour = "white", size = 0.7) +
  labs(fill = "Population Aged 20 - 84")
```



Spatial Analysis

Data Preparation

First we take the population counts for each census tract and convert to the appropriate projected coordinate system. When doing this, we localize each tract level population at the census tract centroid.

```
#We extract the population counts ONLY for the ACS year of interest
stl_boundaries <- get_acs(geography = "block group"
                      , state = c("MO")
                      ,
                      county = c("St. Louis city")
                      ,
                      variables = c("B01001_001")
                      ,
                      geometry = TRUE,
                      year = 2019)
stl_count <- data.frame(data.table(stl_value)
                        [, estimate := sum(estimate), by = GEOID])
stl_poly <- st_as_sf(stl_count)

stl_centroid <- stl_poly %>%
  st_transform(3857) %>% # convert to projected coord system for better centroid
  st_centroid()
```

```
participants_centroid <- participants_poly %>%
  st_transform(3857) %>% # convert to projected coord system for better centroid
  st_centroid()

#then we apply appropriate coordinate reference system transformation
stl_boundaries <- st_transform(stl_boundaries$geometry, crs = 3857)


#Here we define the overall city limits as the operating window used
#throughout these calculations
stl_owin <- as.owin(as_Spatial(stl_boundaries))

#applying appropriate coordinate reference system transformation
#to the census level data
tmp <- st_transform(stl_centroid$geometry, crs = 3857)
xy <- data.frame(st_coordinates(tmp))


stl_centroid_df <- data.frame("x" = rep(xy$X, times = stl_centroid$estimate),
                              "y" = rep(xy$Y, times = stl_centroid$estimate))

#Similarly, we generate a count of the number of participants in each
#census tract, then locate each of them at the centroid of the tract
tmp <- st_transform(participants_centroid$geometry, crs = 3857)
xy <- data.frame(st_coordinates(tmp))

participant_centroid_df <- data.frame("x" = rep(xy$X,
                                                times = participants_centroid$N),
                                      "y" = rep(xy$Y,
                                                times = participants_centroid$N))


#After creating these centroid-based dataframes,
#we have to convert the dataframes to PPP objects.
#We define the window as the city limits of St Louis

PPP_STL_centroids <- as.ppp(stl_centroid_df, W = stl_owin)
PPP_centroids <- as.ppp(participant_centroid_df, W = stl_owin)
```

Cramer-von Mises test for observed vs predicted distribution of spatial values

The two-sample Cramer-von Mises test tests for differences between the spatial distributions of two populations. It is designed to be sensitive to differences in the way populations are distributed across the study area (and you must evaluate within the same study area for both populations), but it is insensitive to differences in abundance between the two populations. Recall that St. Louis City has a population of approximately 300,000 individuals and our sample is 239 scanned individuals. This mismatch in N's is not problematic for this test statistic.

```
# https://rdrr.io/cran/ecespa/man/syrjala.html
#https://esajournals.onlinelibrary.wiley.com/doi/abs/10.2307/2265656

nperm <- 1000
```

```
tmp <- data.frame(data.table(stl_centroid_df)[, .N, by = list(x, y)])
tmp2 <-data.frame(data.table(participant_centroid_df)[, .N, by = list(x, y)])


syr_frame <- data.frame("x" = c(tmp$x,
                                tmp2$x),
                        "y" = c(tmp$y,
                                tmp2$y),
                        "var1" = c(tmp$N, rep(0, nrow(tmp2))),
                        "var2" = c(rep(0, nrow(tmp)), tmp2$N))

full_sample <- data.table(syr_frame)[, .(sum(var1), sum(var2)), by = list(x, y)]


coords <- data.frame(full_sample[, c("x", "y")])
res <- syrjala(coords, full_sample$V1, full_sample$V2, nperm = nperm)
psi <- res$cvm.obs #psi
pvalue <- (sum(res$cvm.sim >=psi)+1)/nperm

rm(syr_frame)
```

We observe no difference in spatial distributions between the sample and population based on the 1000 permutation Cramer-von Mises or Kolmogorov-Smirnov test

```
res
```

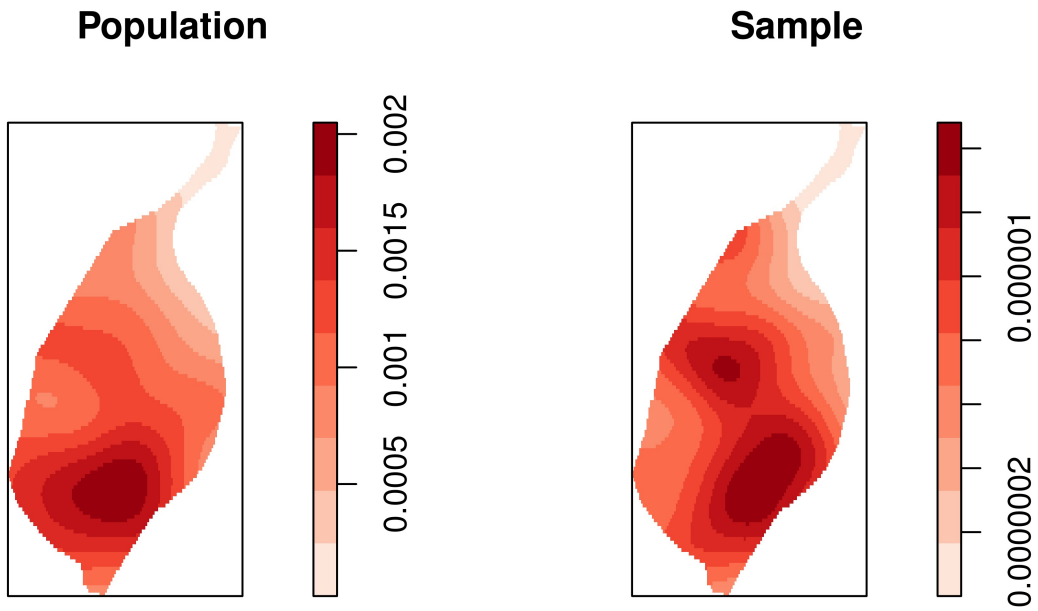
```
## Cramer-von Misses test for the difference between
## the spatial distributions of  full_sample$V1 and full_sample$V2
## based on 1000 permutations.
##
##    psi:     0.4944819
##    p-value: 0.2197802
##
## Kolmogorov-Smirnov test for the difference between
## the spatial distributions of  full_sample$V1 and full_sample$V2
## based on 1000 permutations.
##
##    psi:     0.0954775
##    p-value: 0.3446553
```

Generating Figures 1C and 1D

Spatstat has built in functionality to generate kernel density plots

```
grRd <- colorRampPalette(brewer.pal(7, "Reds"))(10-1)
#Define legend stretch for colors
co_pop <- colourmap(grRd, range = c(min(density(PPP_STL_centroids)),
                                    max(density(PPP_STL_centroids))))
co_samp <- colourmap(grRd, range = c(min(density(PPP_centroids)),
                                     max(density(PPP_centroids))))

par(mfrow=c(1, 2))
```

```
plot(density(PPP_STL_centroids), col = co_pop, main = "Population",
     sub = "Kernel Density")
plot(density(PPP_centroids), col = co_samp, main = "Sample",
     sub = "Kernel Density")
```



Probability Map Generation (Figure 1E)

The kernel ratio of the intensity of cases vs. controls can be used to show the spatial variation of relative risk of developing high BAG. Here we apply a Monte Carlo simulation to estimate the case vs. control intensity across the city.

First we do some preparation of the Point Pattern objects

```
#Define what is a "high BAG"
participants$marks <- ifelse(participants$T1BrainAgeGap >
                               mean(participants$T1BrainAgeGap) +
                               1.5 * sd(participants$T1BrainAgeGap),
                                                    "HighBAG", "control")


#convert dataframe to proper data structure
tmp <- SpatialPoints(participants[, c("lon", "lat")],
                     proj4string=CRS("+proj=longlat"))
tmp.UTM <- spTransform(tmp, CRS("+init=epsg:3857"))
PPP <- as.ppp.SpatialPoints(tmp.UTM)
```

```
PPP$marks <- as.factor(participants$marks)
marks(PPP) <- relevel(PPP$marks, "control") #Setting controls as reference
PPP$window <- stl_owin

rm(stl_owin)
```

Smoothing bandwidths can be chosen manually or via cross-validation. There are several methods supported by R. We looked at both the Diggle and Cronie & van Lieshout's Criterion and then manually chose a smoothing value in between those two.

```
bw.diggle(PPP)
```

```
##    sigma
## 498.8473
```

```
bw.CvL(PPP)
```

```
##    sigma
## 1715.587
```

```
bw_est <- 1000 #hand chosen bandwidth. used as a smoothing parameter.
```

Then we can calculate the kernel density comparing cases (individuals with high BAG) to controls

```
  cases<-unmark(subset(PPP, marks(PPP) =="HighBAG"))
  ncases<-npoints(cases)
  controls<-unmark(subset(PPP, marks(PPP) =="control"))
  ncontrols<-npoints(controls)


  kcases<-density(cases, bw_est)
  kcontrols<-density(controls, bw_est)

  #dropping missing cells
  spkratio0<-as(kcases, "SpatialGridDataFrame")
  names(spkratio0)<-"kcases"
  spkratio0$kcontrols<-as(kcontrols, "SpatialGridDataFrame")$v
  spkratio<-as(spkratio0, "SpatialPixelsDataFrame")

  spkratio$kratio <- spkratio$kcases/spkratio$kcontrols
  spkratio$logratio <- log(spkratio$kratio)-log(ncases/ncontrols)

  rr<-relrisk(PPP, bw_est) #calculate relative risk for probability plot
  spkratio$prob<-as(as(rr, "SpatialGridDataFrame"),
                        "SpatialPixelsDataFrame")$v #store for probability plot

niter <- 99
  ratio <- rep(NA, niter)
  pvaluemap <- rep(0, nrow(spkratio))
  rlabelratio <- matrix(NA, nrow=niter, ncol=nrow(spkratio))
```

```
  #MC test script
  set.seed(1)
  for(i in 1:niter){
    PPP0<-rlabel(PPP)
    casesrel <- unmark(subset(PPP0, marks(PPP0) =="HighBAG"))
    controlsrel <- unmark(subset(PPP0, marks(PPP0) =="control"))

    kcasesrel <- density(casesrel, bw_est) #calculating density
    kcontrolsrel <- density(controlsrel, bw_est) #calculating density
    #calculating the ratio of the two densities
    kratiorel <- eval.im(kcasesrel/kcontrolsrel)
    tryCatch(
      expr = {
        rlabelratio[i,] <- as(as(kratiorel, "SpatialGridDataFrame"),
                                        "SpatialPixelsDataFrame")$v
        pvaluemap <- pvaluemap + (spkratio$kratio < rlabelratio[i,])


      }, error = function(e){
        rlabelratio[i,] <- NA
        pvaluemap <- NA
      }
    )
  }

  #Calculating the kernel ratio
  cellsize<-kcontrols$xstep*kcontrols$ystep
  ratiorho <- cellsize*sum((spkratio$kratio-ncases/ncontrols)^2)
  ratio <- cellsize*apply(rlabelratio, 1,
                          function(X, rho0 ){sum((X-rho0)^2)},
                          rho0=ncases/ncontrols


  )

    pvaluerho <- (sum(ratio > ratiorho, na.rm = TRUE)+1)/(niter+1)
```

The results of this Monte Carlo simulation result in a spatial distribution of the relative intensity of cases. We can also extract p values from the simulation in order to identify the area where one is statistically significantly more likely to have high BAG.

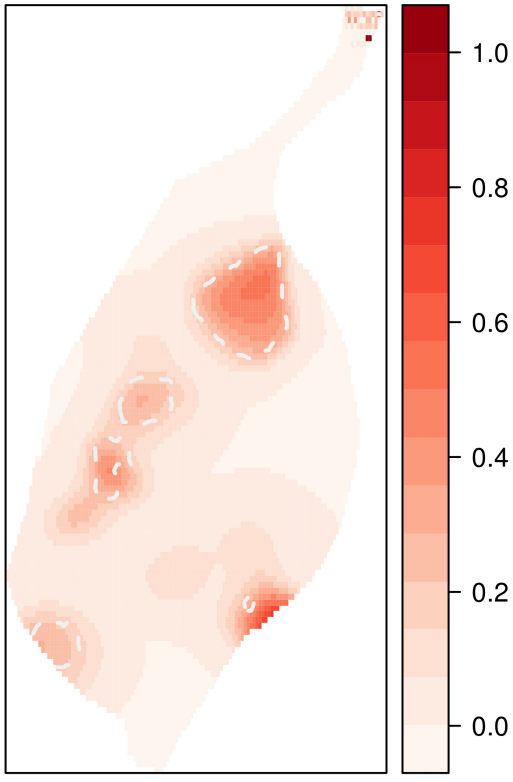
```
alpha <- 0.05 #Pvalue
alpha_flip <- 1 - alpha

  spkratio$pvaluemap <- (pvaluemap+1)/(niter+1)



  imgpvalue <- as.image.SpatialGridDataFrame(spkratio["pvaluemap"])
  clpvalue <- contourLines(x = imgpvalue$x, y = imgpvalue$y, z = imgpvalue$z,
                           levels=c(0,as.numeric(alpha),
                                    as.numeric(alpha_flip), 1))
  cl <- ContourLines2SLDF(clpvalue)
```

```
  cl05 <- cl[cl$level == alpha,]
  xzx <- slot(slot(cl05, "lines")[[1]], "Lines")
  cl05a <- SpatialLines(list(Lines(xzx, ID=alpha)))
  lyt05 <- list("sp.lines", cl05a, lwd=2, lty=2, col="grey95")
  brks <- quantile(spkratio$kratio[spkratio$kratio>0], seq(0,1,1/10),
                   na.rm=TRUE)
  brks[1] <- 0
```

Generating Figure 1F

Following this simulation, we apply a binary regression estimator to assess the probability of being a case at every grid cell in the study region. This will yield a map that shows the probability of having high BAG across St. Louis.

```
ats <- seq(0,max(spkratio$prob),length.out=17)
  cols <- colorRampPalette(brewer.pal(8, "Reds"))(length(ats)-1)

    print(spplot(spkratio, "prob",
                      col.regions=cols,
                sp.layout=list(lyt05)
  ), silent = TRUE)
```

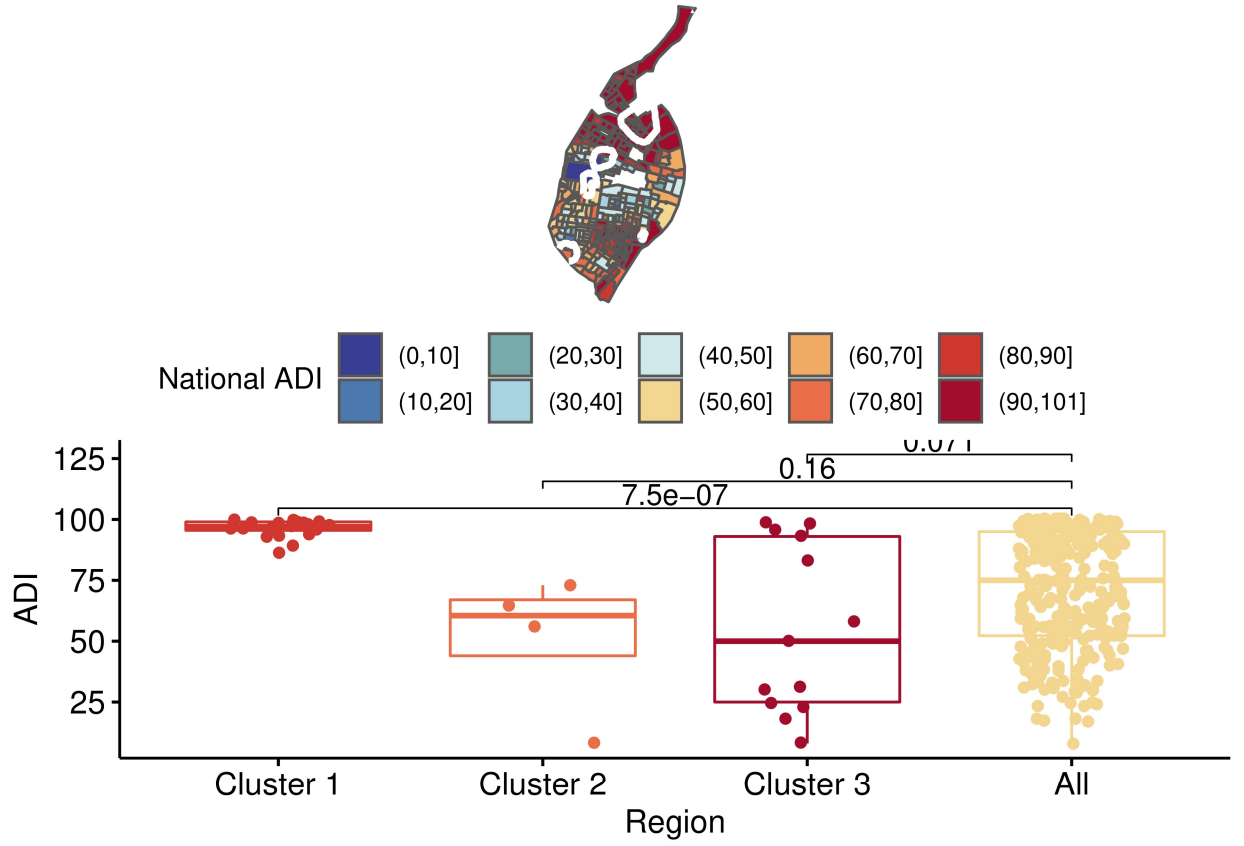


Generating Supplemental Figure 3

We wanted to compare the ADI of the identified regions with the typical ADI of St. Louis, MO. In order to do this, we needed to find the spatial intersection of the white lines from Figure 1F and the tract-level ADI from Figure 1A.

```
#we use ggfortify to get the geometry of the white lines
mySLDF_fortify <- fortify(lyt05[[2]]@lines[[1]])
#apply crs transformation to existing boundaries
stl_boundaries_trans <- sf::st_transform(plot_df, crs(stl_boundaries))

#Then we have to separate each identified area and
#find the intersection of the circle with the known census tracts

# Cluster1
cl05a1 <- SpatialLines(list(Lines(xzx[[4]], ID=alpha)))
lyt051 <- list("sp.lines", cl05a1, lwd=2, lty=2, col="grey95")
line_trans1 <- st_as_sf(lyt051[[2]])
line_trans1 <- st_polygonize(line_trans1)
result21 <- st_intersection(st_set_crs(line_trans1,
                                       crs(stl_boundaries)),
                            stl_boundaries_trans)

#Cluster2
cl05a2 <- SpatialLines(list(Lines(xzx[[2]], ID=alpha)))
lyt052 <- list("sp.lines", cl05a2, lwd=2, lty=2, col="grey95")
line_trans2 <- st_as_sf(lyt052[[2]])
line_trans2 <- st_polygonize(line_trans2) #This closes an open curve
result22 <- st_intersection(st_set_crs(line_trans2, crs(stl_boundaries)),
                                                    stl_boundaries_trans)

#Cluster2
cl05a3 <- SpatialLines(list(Lines(xzx[[3]], ID=alpha)))
lyt053 <- list("sp.lines", cl05a3, lwd=2, lty=2, col="grey95")
line_trans3 <- st_as_sf(lyt053[[2]])
line_trans3 <- st_polygonize(line_trans3)
result23 <- st_intersection(st_set_crs(line_trans3, crs(stl_boundaries)),
                                                   stl_boundaries_trans)

#Then we generate the dataframe necessary to generate the boxplot
ADI_boxplot <- data.frame("Region" = c(rep("Cluster 1",
                                           length(result21$ADI_NATRANK)),
                                       rep("Cluster 2",
                                           length(result22$ADI_NATRANK)),
                                       rep("Cluster 3",
                                           length(result23$ADI_NATRANK)),
                                      rep("All",
                                          length(stl_boundaries_trans$ADI_NATRANK))),
                          "ADI" = c(result21$ADI_NATRANK, result22$ADI_NATRANK,
                                    result23$ADI_NATRANK,
                                    stl_boundaries_trans$ADI_NATRANK))

p1 <- ggplot(data = stl_boundaries_trans[!is.na(stl_boundaries_trans$ADI_cut),]) +
  geom_sf(aes(fill = ADI_cut)) +
```

```
  #Manually selecting hex colors that match those used on UW's site
  scale_fill_manual(values = c("#3a3f94", "#4c78b0",
                               "#77aaab",
                               "#a8d2e0", "#cfe9ea", "#f3d890",
                               "#f2ab65", "#ea704a", "#d03830",
                               "#a30d2f"), name = "National ADI") +
  theme_void() + theme(legend.position = "bottom") +
  geom_path(data = mySLDF_fortify, aes(x=long, y=lat, group=group), colour = "white", size = 1.2)

my_comparisons <- list( c("Cluster 1", "All"), c("Cluster 2", "All"), c("Cluster 3", "All") )


p2 <- ggboxplot(ADI_boxplot, x = "Region", y = "ADI",
          color = "Region", palette = "jco", add = "jitter")+
   scale_color_manual(values = c("#d03830",  "#ea704a", "#a30d2f", "#f3d890" ))+
  stat_compare_means(comparisons = my_comparisons) +
  theme(legend.position = "none")

grid.arrange(p1, p2, nrow = 2, ncol = 1)
```




```
rm(my_comparisons, ADI_boxplot)
```