A Practitioner's Guide to Geospatial Analysis

6/15/2022

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")
```

```
## 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)
    }
  }
)
```

Statistical Analysis for Demographic 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 from the actual results presented in the manuscript

```
#reading in prepped data
participants <- readRDS(".././Data/Synthetic_Participant_Data.RDS")
```

Population Estimate Extraction from ACS

Here we extract American Community Survey (ACS) counts of relevant demographic variables at the census tract level in St. Louis, MO.

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

```
census_api_key(XXX, overwrite = TRUE, install = TRUE)
```

After establishing your relationship with the census API, 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 = 2019,
                       dataset = "acs5",
                       cache = TRUE)
```

You can view the list of all variables if you want, or you can use the demographic variables that we also evaluate in the corresponding manuscript

```
View(vars)
```

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

- B00001_001 - population by tract
- B01002_001 - estimated median age
- B01001_002 - total num of males

- B01001_026 - total num of females
- B02001_002 - total num of whites
- B02001_003 - total num of blacks
- B16010_002 – total less than high school graduate
- B16010_015 – high school graduate
- B16010_028 – some college
- B16010_041 – BS or higher

```
stl_value <- get_acs(geography = "tract"
                     , state = c("MO")
                     ,
                     county = c("St. Louis city")
                     ,
                     variables = c("B01001_001", "B01002_001", "B01001_002",
                         "B01001_026", "B02001_002", "B02001_003",
                         "B16010_002", "B16010_015", "B16010_028", "B16010_041")
                     ,
                     geometry = TRUE,
                     year = 2019)
```

We then rename the variables for convenience/interpretability and use data.table to summarize each of these values.

```
stl_value$variable <- as.factor(stl_value$variable)
levels(stl_value$variable) <- list(population = "B01001_001",
                                   age_medianAge = "B01002_001",
                                   sex_malesCount = "B01001_002",
                                   sex_femalesCount = "B01001_026",
                                   race_whiteCount = "B02001_002",
                                   race_blackCount = "B02001_003",
                                   educ_lessThanHS = "B16010_002",
                                   educ_HS = "B16010_015",
                                   educ_someCollege ="B16010_028",
                                   educ_BSorMore = "B16010_041")

STL_tableone_characteristics <- data.table(data.frame(stl_value))[, .(mean = mean(estimate),
                                                                       median = median(estimate),
                                     sum = sum(estimate),
                                     sd = sd(estimate)), by = list(variable)]
```

And finally we compare the ACS data to our sample.

```
Table1 <- data.frame("Characteristic" = c("N", "Age (Mean)", "Sex - Female",
                                  "Sex - Male", "Race - Black", "Race - White",
                                  "Education - less than high school",
                                  "Education - high school", "Education - some college",
                                  "Education - BS or higher"),
           "STL_City" = c(as.numeric(STL_tableone_characteristics
                                     [STL_tableone_characteristics$variable == "population", "sum"]),
                          as.numeric(STL_tableone_characteristics
                                     [STL_tableone_characteristics$variable == "age_medianAge", "mean"]),
                          as.numeric(STL_tableone_characteristics
                                     [STL_tableone_characteristics$variable == "sex_malesCount", "sum"]),
```

```
                          as.numeric(STL_tableone_characteristics
                                     [STL_tableone_characteristics$variable == "sex_femalesCount", "sum
                          as.numeric(STL_tableone_characteristics
                                     [STL_tableone_characteristics$variable == "race_blackCount", "sum"]
                          as.numeric(STL_tableone_characteristics
                                     [STL_tableone_characteristics$variable == "race_whiteCount", "sum"]
                          as.numeric(STL_tableone_characteristics
                                     [STL_tableone_characteristics$variable == "educ_lessThanHS", "sum"]
                          as.numeric(STL_tableone_characteristics
                                     [STL_tableone_characteristics$variable == "educ_HS", "sum"]),
                          as.numeric(STL_tableone_characteristics
                                     [STL_tableone_characteristics$variable == "educ_someCollege", "sum
                          as.numeric(STL_tableone_characteristics
                                     [STL_tableone_characteristics$variable == "educ_BSorMore", "sum"])
                          ),
          "Participants" = c(length(unique(participants$related_study_id)),
                             mean(participants$TrueAge),
                             nrow(participants[participants$Sex == "male",]),
                             nrow(participants[participants$Sex == "female",]),
                             nrow(participants[participants$Race == "Black",]),
                             nrow(participants[participants$Race == "White",]),
                             nrow(participants[participants$Education < 12, ]),
                             nrow(participants[participants$Education == 12, ]),
                             nrow(participants[participants$Education > 12 & participants$Education <
                             nrow(participants[participants$Education >=16, ]))
          )

Table1
```

```
##                       Characteristic     STL_City Participants
## 1                                  N 308174.00000       0.0000
## 2                          Age (Mean)     37.27925      53.1943
## 3                        Sex - Female 149175.00000     152.0000
## 4                          Sex - Male 158999.00000      87.0000
## 5                        Race - Black 143018.00000     146.0000
## 6                        Race - White 143401.00000      93.0000
## 7  Education - less than high school  26828.00000      39.0000
## 8             Education - high school  52444.00000      60.0000
## 9           Education - some college  60331.00000      58.0000
## 10          Education - BS or higher  79669.00000      82.0000
```

We apply a t-test to continuous variables (Age) and a chi square test to categorical variables (race, sex, education) to test for demographic differences. We find our sampled cohort to be equivalent on the basis of years of education, but to be older, more heavily male, and containing more Black individuals than would be expected based on the city population.

Assessing cohort differences by age

```
t_test <- function (mean1, mean2, sd1, sd2, n1, n2){
  t <- (mean1 - mean2) / sqrt((sd1)^2 / n1 + (sd2)^2 / n2)
  p <- 2*pt(q = abs(t), df = (n1 + n2 - 2), lower.tail = FALSE) #2 tailed t test
```

```
  return(list(t, p))
}

t_score <- t_test(37.3, 53.2, 6.4, 11.5, 308174, 239)

print(paste0("T = ", t_score[[1]], " p = ", t_score[[2]]))
```

```
## [1] "T = -21.3720451385086 p = 2.88226745979766e-101"
```

Assessing cohort difference by sex

```
sex_chi <- data.frame ("STL_City" = c(as.numeric(STL_tableone_characteristics[STL_tableone_characteristi
                                      as.numeric(STL_tableone_characteristics[STL_tableone_characteris
                       "Cohort" = c(nrow(participants[participants$Sex == "male",]),
                                    nrow(participants[participants$Sex == "female",])))

chisq.test(sex_chi)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  sex_chi
## X-squared = 21.466, df = 1, p-value = 3.602e-06
```

Assessing cohort difference by race

```
race_chi <- data.frame ("STL_City" = c(as.numeric(STL_tableone_characteristics[STL_tableone_characteris
                                       as.numeric(STL_tableone_characteristics[STL_tableone_characteristi
                        "Cohort" = c(nrow(participants[participants$Race == "Black",]),
                                     nrow(participants[participants$Race == "White",])))

chisq.test(race_chi)
```

```
##
##  Pearson's Chi-squared test with Yates' continuity correction
##
## data:  race_chi
## X-squared = 11.443, df = 1, p-value = 0.0007175
```

Assessing cohort difference by educational level

```
ed_chi <- data.frame ("STL_City" = c(as.numeric(STL_tableone_characteristics[STL_tableone_characteristi
                                     as.numeric(STL_tableone_characteristics[STL_tableone_characteristi
                                     as.numeric(STL_tableone_characteristics[STL_tableone_characteristi
                                     as.numeric(STL_tableone_characteristics[STL_tableone_characteristi
                      "Cohort" = c(nrow(participants[participants$Education < 12, ]),
```

```
                                    nrow(participants[participants$Education == 12, ]),
                                    nrow(participants[participants$Education > 12 & participants$Educa
                                    nrow(participants[participants$Education >=16, ])))

chisq.test(ed_chi)
```

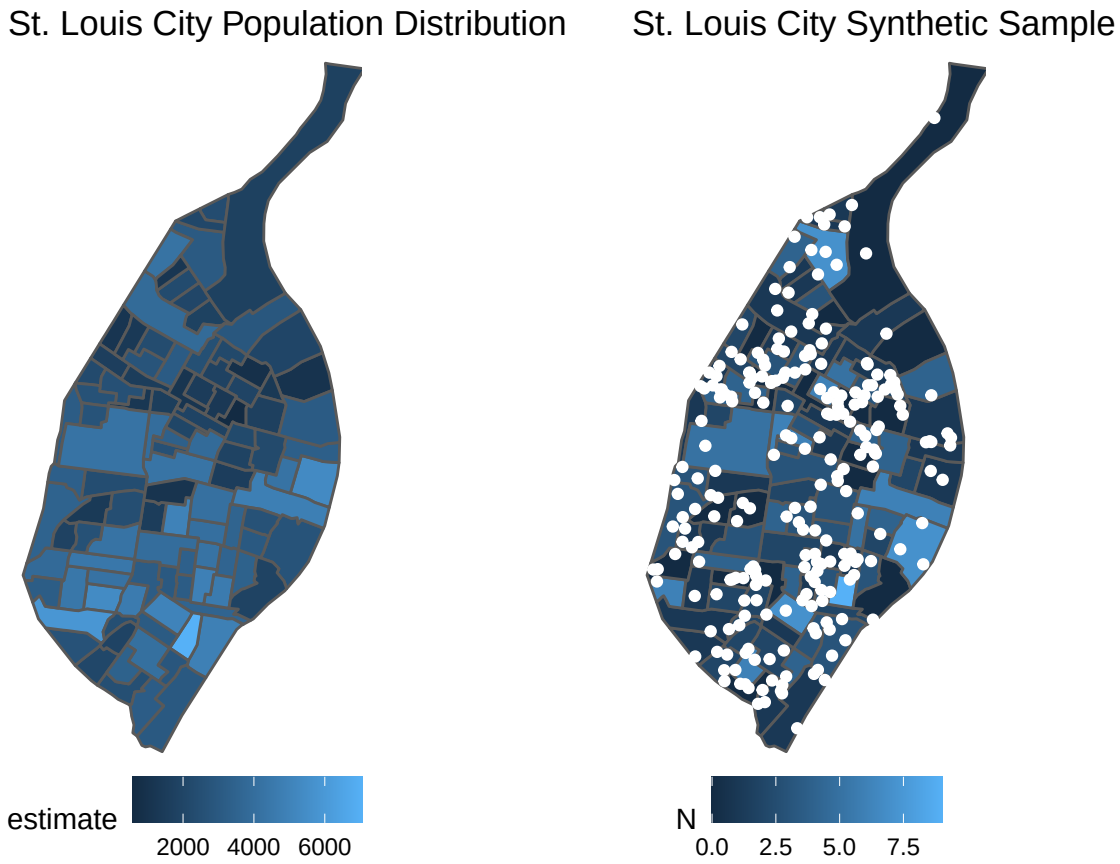
```
##
##  Pearson's Chi-squared test
##
## data:  ed_chi
## X-squared = 4.5762, df = 3, p-value = 0.2056
```

Spatial Analysis

Visualizing participant distribution vs. Overall City Population

We present the sampled distribution overlain on tract level sampling information. This is displayed in conjunction with the actual ACS tract level counts.

```
#getting participant count by census tract
participants_count <- data.frame(data.table(participants)[, .N, by = GEOID])
participants_poly <- merge(participants_count,
                           stl_value[, 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)

p1 <- ggplot(stl_value[stl_value$variable == "population",]) +
  aes(fill = estimate) + geom_sf() +
  xlim(c(-90.32, -90.175)) + ylim(c(38.54, 38.77)) + theme_void() +
  theme(legend.position = "bottom")  +
  ggtitle("St. Louis City Population Distribution")

p2 <- ggplot(participants_poly) +
  aes(fill = N) + geom_sf() +
  xlim(c(-90.32, -90.175)) + ylim(c(38.54, 38.77)) + theme_void() +
  geom_point(data = participants, aes(x = lon, y = lat, fill = 1), colour = "white") +
  theme(legend.position = "bottom")  +
  ggtitle("St. Louis City Synthetic Sample Distribution")

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



Kernel Density Map Generation

The Kernel Density Map shows the continuous distribution of city residents or participants across the sampled area. We can generate this map and identify "hot spots" of high population concentration. There are empirical ways to test for differences in spatial distribution between the sample and population, but generating a kernel density map allows for a first pass visualization of this data.

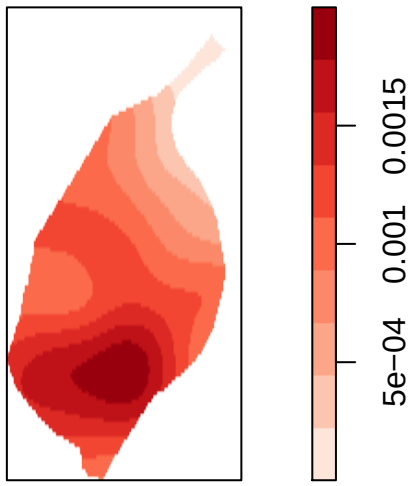
First we take the population counts for each census tract and convert to the appropriate projected coordinate system. When doing this, we localize each tract level population at the census tract centroid.

```
#We extract the population counts ONLY for the ACS year of interest
stl_boundaries <- get_acs(geography = "tract"
                      , state = c("MO")
                      ,
                      county = c("St. Louis city")
                      ,
                      variables = c("B01001_001")
                      ,
                      geometry = TRUE,
                      year = 2019)
stl_count <- data.frame(data.table(stl_value[stl_value$variable == "population",])
                        [, 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
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)
```

Then we can use the built in spatstat functionality to generate kernel density plots

```
grRd <- colorRampPalette(brewer.pal(7, "Reds"))(10-1)
co_pop <- colourmap(grRd, range = c(1e-06, 2e-03))  #Define legend stretch for colors


co_samp <- colourmap(grRd, range = c(1e-07, 2e-05))

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

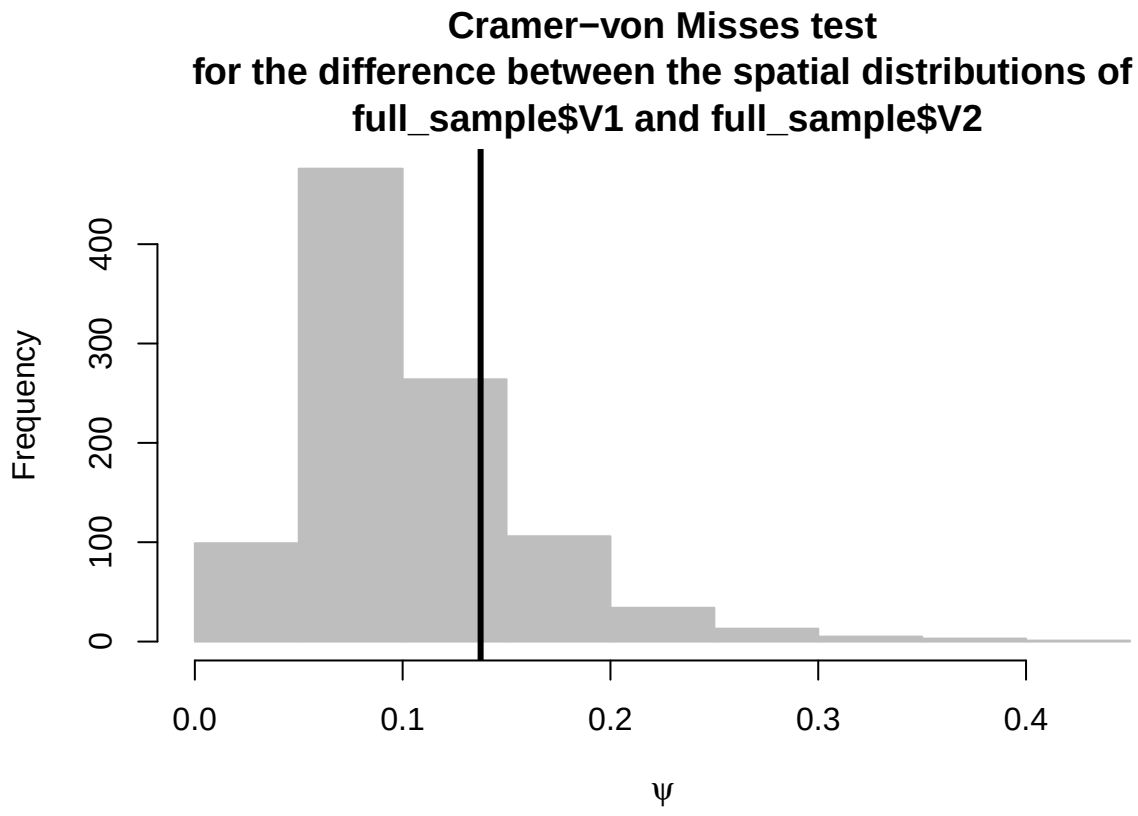


Cramer-von Mises test for observed vs predicted distribution of spatial values

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

plot(res)
```



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:     0.1375183
##    p-value: 0.2097902
##
## Kolmogorov-Smirnov test for the difference between
## the spatial distributions of  full_sample$V1 and full_sample$V2
## based on 1000 permutations.
##
##    psi:     0.08582897
##    p-value: 0.2967033
```

K Function to Test for Equal Clustering

Ripley's K describes the clustering pattern of a set of points. This is a second order property that describes the interaction between multiple points. Specifically, the K function measures the number of events occurring within a given distance of a particular event.

Here we apply the spatstat implementation to calculate the difference of two K functions. We test for differences in clustering between participants with high brain age gap (BAG) and the remainder of our participants (controls).

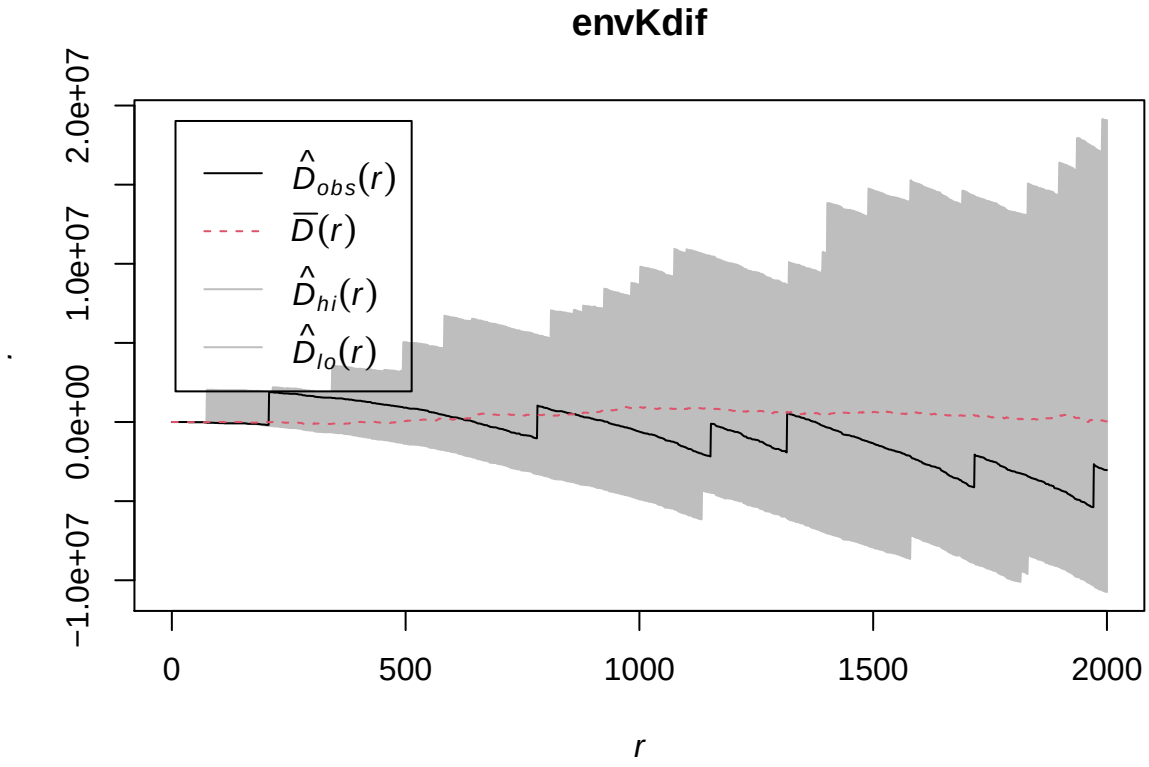
Testing for differences in spatial distribution between high BAG and controls

Here we apply the default spatstat approach to calcuating the difference in K functions. We are also able to use actual lat/lon coordinates for each participant since we are using sample data rather than ACS data. We see that the black line indicating the K function for participants with high BAG lies fully within the envelope, indicating no difference in clustering patterns between the participants with high BAG and controls.

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

#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

#Then calculate the K difference envelope using standard spatstat method:
 r<-seq(0,2000, by = 1)

 Kdif<-function(Xppp, r, case_mark_name, control_mark_name, cr="border")
{
  k1<-Kest(Xppp[marks(Xppp)==case_mark_name], r=r, correction=cr)
  k2<-Kest(Xppp[marks(Xppp)==control_mark_name], r=r, correction=cr)

  res<-data.frame(r=r, D=k1[[cr]]-k2[[cr]])
  return(fv(res, valu="D", fname="D"))
}


 envKdif<-envelope(PPP, Kdif, r=r, nsim=99, case_mark_name = "HighBAG", control_mark_name = "control",
                   savefuns=TRUE,
                   simulate=expression(rlabel(PPP)))
```

```
## Generating 99 simulations by evaluating expression  ...
## 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28
## 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 6
## 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98,  99.
##
## Done.
```

```
plot(envKdif)
```



Probaility Map Generation

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.

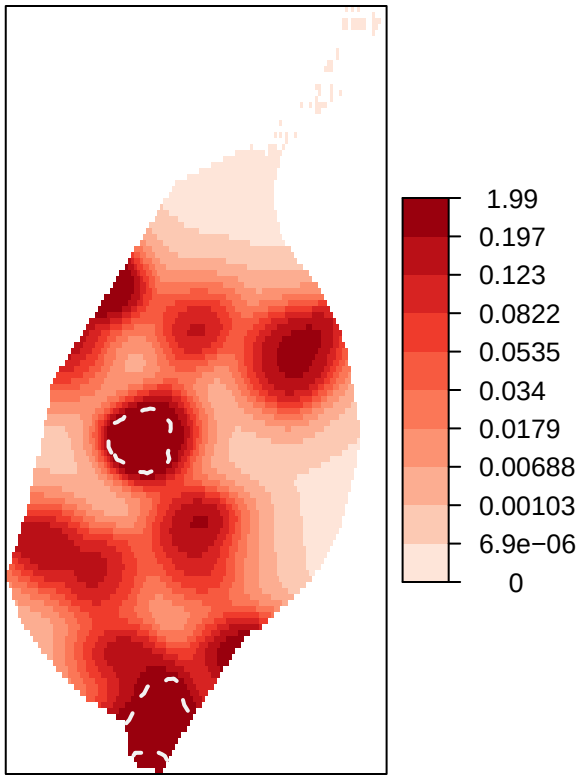
```
bw_est <- 1000 #hand chosen bandwidth. used as a smoothing parameter.
cases<-unmark(subset(PPP, marks(PPP) =="HighBAG"))
ncases<-npoints(cases)
controls<-unmark(subset(PPP, marks(PPP) =="control"))
ncontrols<-npoints(controls)

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

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
    kratiorel <- eval.im(kcasesrel/kcontrolsrel) #calculating the ratio of the two densities
    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 can be used to generate a plot 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. This is shown with a white dashed line.

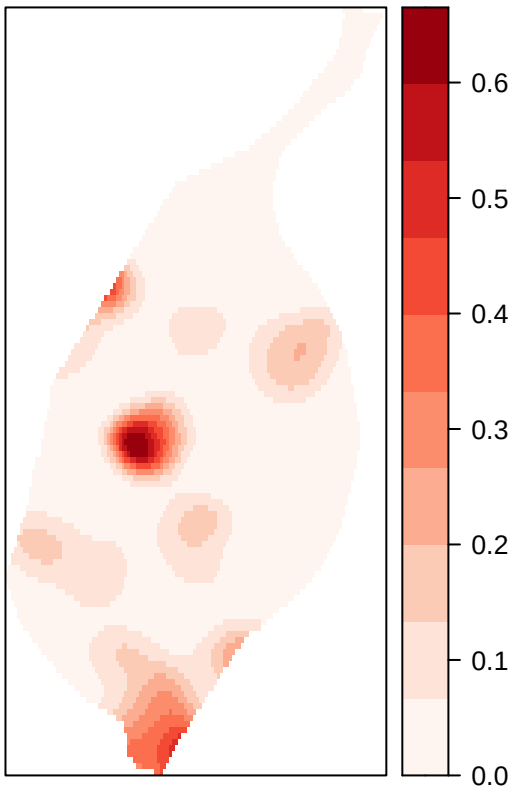
```
alpha <- 0.05 #Pvalue
alpha_flip <- 1 - alpha

  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
lbrks <- formatC(brks, 3, 6, "g", " ")
cols <- colorRampPalette(brewer.pal(7, "Reds"))(length(brks)-1)
colorkey<-list(labels=lbrks,
               at=(0:10)/10, height=.5)

print(spplot(spkratio, "kratio",
                   col.regions=cols,
                   do.log=TRUE,
                   colorkey=colorkey,
                   at=c(0, brks[-c(1,11)], max(spkratio$kratio, na.rm=TRUE)),
             sp.layout=list(lyt05)
), silent = TRUE)
```



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=11)
  cols <- colorRampPalette(brewer.pal(8, "Reds"))(length(ats)-1)
  p <- print(spplot(spkratio, "prob", col.regions=cols, at=ats))
```



Supplemental Materials

ADI Map Generation

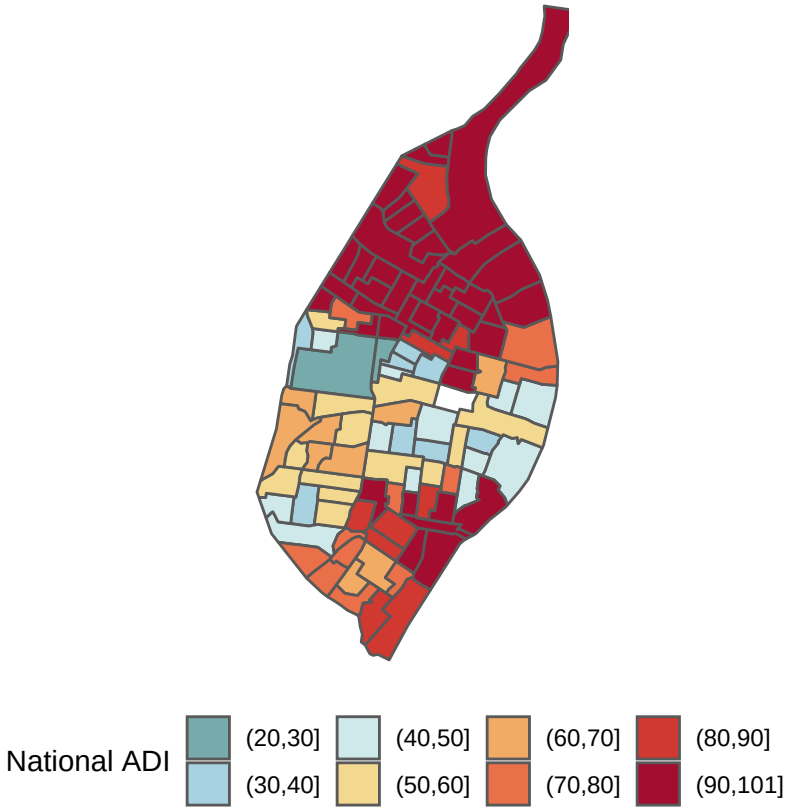
The University of Wisconsin offers neighborhood level estimates of area deprivation (https://www.neighborhoodatlas.medicine.wisc.edu/). These tract-level values are freely available for download. Here we demonstrate an R-based way to replicate the Arc-GIS based mappings that are displayed on the University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

Note that conversion across various types of geographic identifiers can sometimes be cumbersome. We recommend https://www.nhgis.org/geographic-crosswalks for assistance when trying to figure out which numeric code you are using and how to translate it to another useful code.

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


#Just using ACS data to extract the geometries
#Limiting extracted geometry to St. Louis City Limits
metro_region <- get_acs(geography = "tract"
                        , state = c("MO")
                        ,
                        county = c("St. Louis city")
                        ,
                        variables = "B19013_001"
                        ,
                        geometry = TRUE)


#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) - 1))
ADI$ADI_NATRANK <- as.numeric(as.character(ADI$ADI_NATRANK))
ADI <- data.table(ADI)[, mean(ADI_NATRANK, na.rm = TRUE), by = GEOID]
colnames(ADI)[2] <- "ADI_NATRANK"

#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)))
```

```r
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")
```


National ADI
(20,30]
(30,40]
(40,50]
(50,60]
(60,70]
(70,80]
(80,90]
(90,101]