Community boundaries

Danielle Medgyesi

12/20/2022

Open Buildings

Full dataset from Open Buildings for the grid cell containing Ghana (0fd)

https://sites.research.google/open-buildings/

Downloaded on 10/28/2021

```
#the full dataset includes >14 million buildings and takes a long time to load
#instead you can read in the filtered data for a chosen district
#outside of our study region to protect the confidentiality of participants

#Buildings<- read_csv("~/GeospatialGhana/Data/0fd_buildings.csv.gz")

#boundaries <- opq(bbox = 'GH-AF') %>%
 # add_osm_feature(key = 'admin_level', value=6) %>%
  #osmdata_sf %>% unique_osmdata

#districts<-boundaries$osm_multipolygons

#D1<- districts %>% filter(name=="Asutifi South District")

#D1<- D1 %>% st_transform(4326)

#first filter buildings inside xy limits
#D_box<- st_bbox(D1)


#BuildingsBound<- Buildings %>%
#  filter(between(longitude, D_box[1], D_box[3])) %>%
 #  filter(between(latitude, D_box[2], D_box[4]))

#BuildingsBound_sf<- st_as_sf(BuildingsBound, wkt = "geometry", crs= 4326)

#BuildingsBound_sf<- st_make_valid(BuildingsBound_sf)

#then filter to buildings inside boundary
#BuildingsBound_sf<- st_filter(BuildingsBound_sf, D1)

#check
#ggplot(data = D1)+geom_sf()+geom_point(data = BuildingsBound_sf, aes(x=longitude,y=latitude))

#st_write(BuildingsBound_sf, "~/GeospatialGhana/Data/BuildingsBound_sf/BuildingsBound_sf.shp")
```

```
BuildingsBound_sf<- st_read("~/GeospatialGhana/Data/BuildingsBound_sf/BuildingsBound_sf.shp",
                            quiet=TRUE)


#as described in manuscript
#we restricted to buildings with a confidence score of 0.7 or greater
#buildings with lower confidence score prone to inaccuracies

BuildingsBound_sf<- BuildingsBound_sf %>% filter(confdnc>=0.7)



colnames(BuildingsBound_sf)
```

```
## [1] "latitud"  "longitd"  "ar_n_mt"  "confdnc"  "fll_pl_"  "geometry"
```

```
#definitions from Open Buildings
#latitud: latitude of the building polygon centroid,
#longitd: longitude of the building polygon centroid,
#ar_n_mt: area in square meters of the polygon,
#confdnc: confidence score [0.5;1.0] assigned by the model,
#geometry: the building polygon in the WKT format (POLYGON or MULTIPOLYGON),
#fll_pl_: the full Plus Code at the building polygon centroid,
```

Map buildings within the region of an example community

```
# Map an example of buildings within community region
#6.863755, -2.446475

Example<- BuildingsBound_sf %>%
  filter(between(latitud, (6.863755-0.01), (6.863755+0.01))) %>%
  filter(between(longitd,  (-2.446475-0.01), (-2.446475+0.01)))
```

In order to access satellite data from Google, users must obtain a valid Google Maps API key

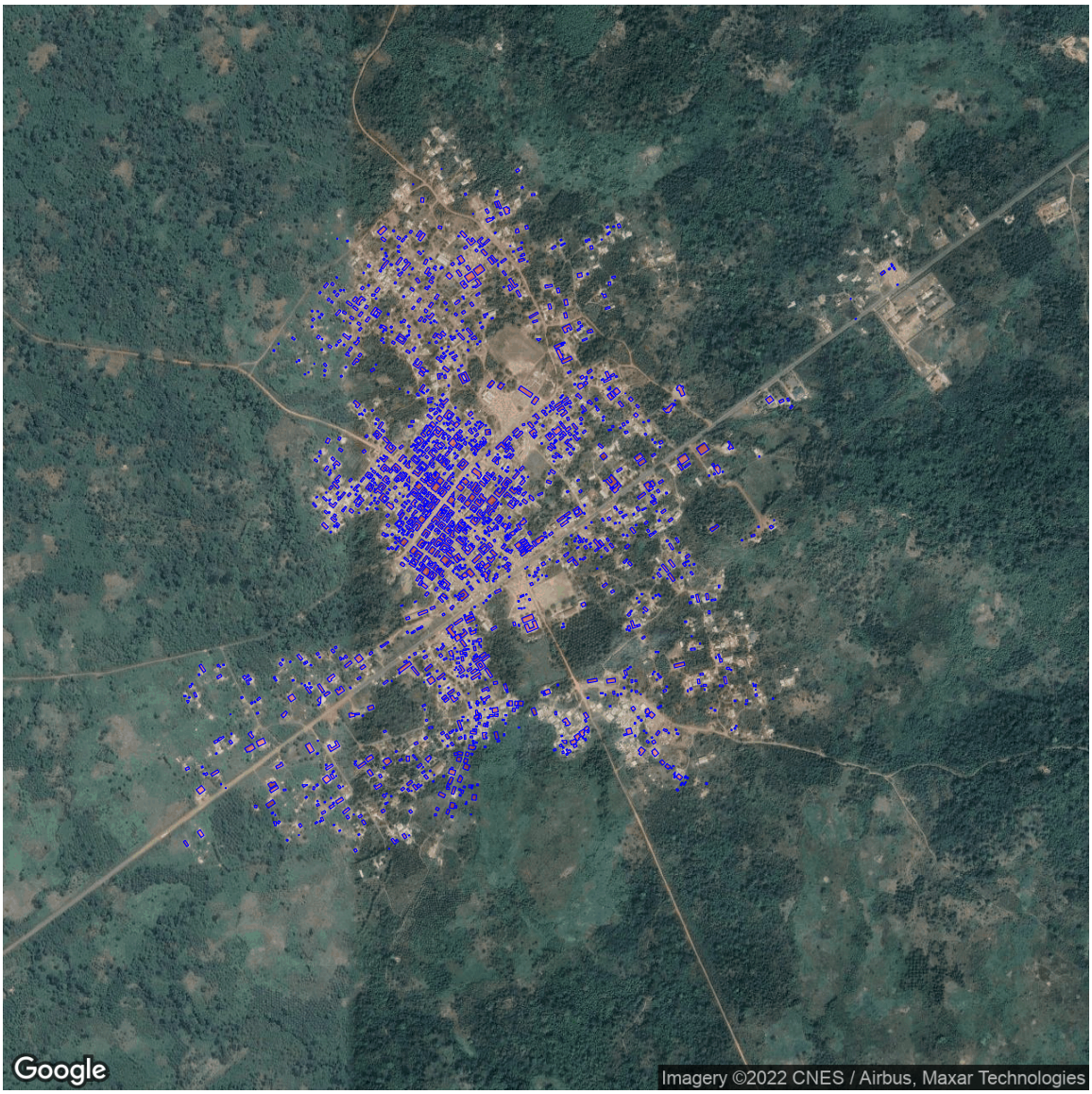
To do so, you can create an account with Google: https://mapsplatform.google.com/

- Begin a new project and create credentials
- I created an API key, selecting "restrict key" > "Maps JavaScript API"
- API can then be called into R using the "register_google" function
- Make sure your API is secure; charges may be apply to high-volume users

If you do not wish to create an API, follow code for plain ggplot maps

```
#map_key <- ""
#register_google(key=map_key)

base<- get_map(location = c(-2.446475, 6.863755), zoom=15, maptype = "satellite")

ggmap(base)+
  coord_sf(crs = st_crs(4326)) +
  geom_sf(data = Example,
```

```
          fill=alpha("#FF0000", .3), color="blue",
          inherit.aes = FALSE)+
  theme(axis.line = element_blank(),
        axis.text = element_blank(),
        axis.ticks = element_blank(),
        plot.margin = unit(c(0, 0, -1, -1), 'lines')) +
  xlab('') +
  ylab('')
```




```
#if you wish to view an interactive map:
#google_map(key = map_key, zoom = 15) %>%
 # add_polygons(data =Example, polyline = "geometry")


#alternatively, use gpplot if no Google API credentials

ggplot()+
```

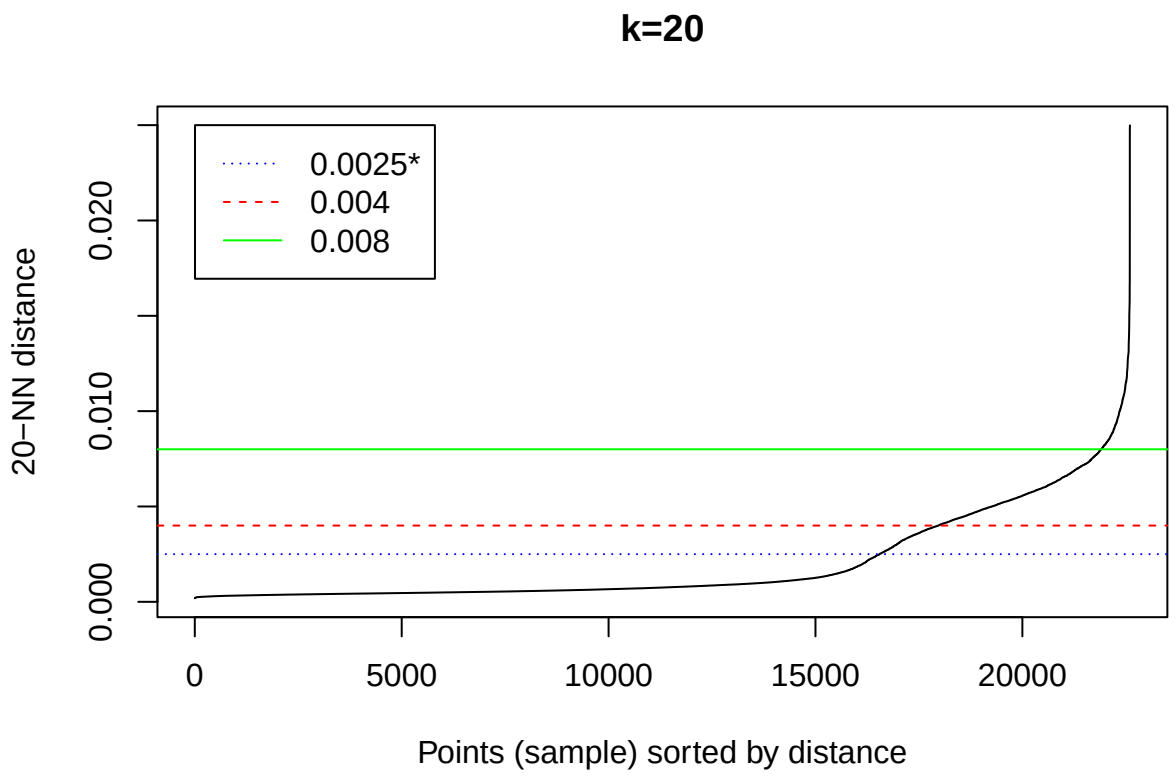
```
geom_sf(data = Example)+
theme_void()
```

k-nearest neighbor graphs

From manucript: Communities were distinguished as groups of buildings belonging to the same cluster. Clusters were identified with the spatial algorithm, density-based spatial clustering of applications with noise (R function dbscan).The DBSCAN method allows for clusters (i.e., communities) to be arbitrary shapes and sizes, does not require that the user to know the number of communities in the dataset and can disregard sparse buildings seemingly not belonging to any community (e.g., outlying or industrial buildings). The algorithm requires two parameters: epsilon (the maximum distance between a given building and neighboring buildings in the same community) and the minimum number of neighboring buildings around each given building in a community. Reasonable epsilon values were selected using k-nearest neighbor graphs (k-NNG), which plot in ascending order the smallest distance between a specified number of neighboring buildings (k) around each building in the dataset. Epsilon values were identified at the elbow of the k-NNG, where distances beyond the elbow suggest that buildings are too far apart to reasonably belong to the same

community. Epsilon values were inputted into the clustering algorithm and further fine-tuned based on visual inspection of the resulting clusters.

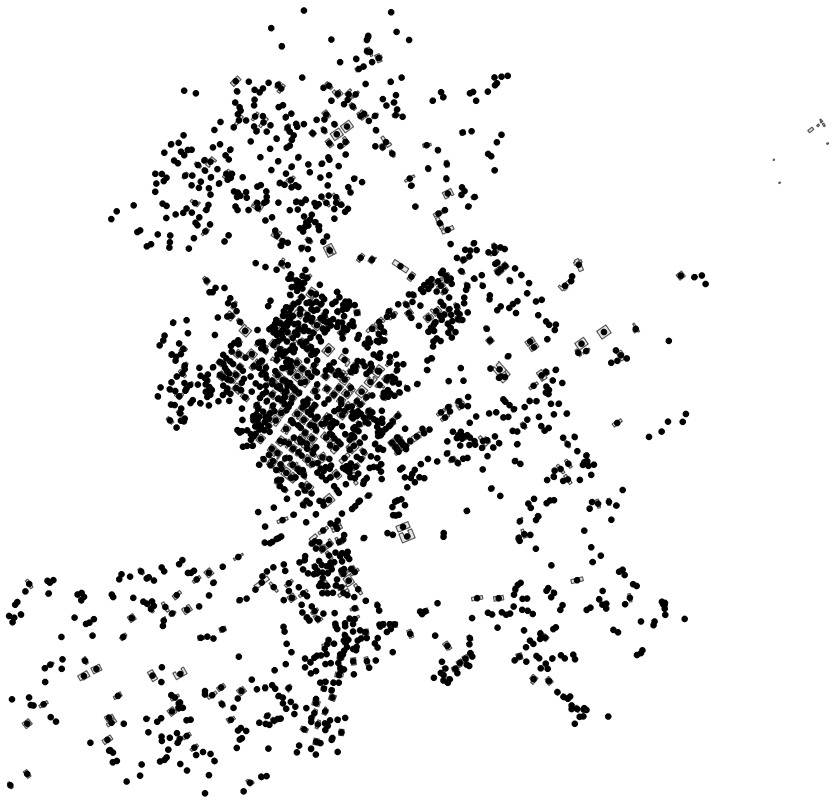
```
#simplify to a data frame removing geometry
#DBSCAN uses the lon/lat (centroid) of each building

BB<-  BuildingsBound_sf %>% st_drop_geometry()

#extract centroid of each building

df<- BB %>% dplyr::select(longitd, latitud)

#note: plot and parameters chosen are different than Figure S2
#since this example is in a district outside of our study region
dbscan::kNNdistplot(df, k = 20)
abline(h = 0.004, col="red", lty = 2)
abline(h = 0.008, col="green", lty = 1)
abline(h = 0.0025, col="blue", lty = 3)
legend(5, 0.025, legend=c("0.0025*", "0.004", "0.008"),
       col=c("blue", "red", "green"), lty=c(3,2,1))
title("k=20")
```



Create community clusters

```
set.seed(123)
```

```r
db <- dbscan::dbscan(df, eps = 0.0025, minPts = 20)


BB_db<- cbind(BB, db$cluster)

BB_db<- BB_db %>% rename(db.cluster= `db$cluster`)%>%
  filter(db.cluster!=0)
#d.b cluster=0 are buildings that do not belong to any community


#building centroids to sf
BB_db = st_as_sf(BB_db, coords = c("longitd", "latitud"),
                 crs = 4326)


#separate each cluster into an element of list
cluster_list<- split(BB_db, f= BB_db$db.cluster)



#example of buildings belonging to the cluster 104
ggplot()+ geom_sf(data = Example)+
  geom_sf(data = cluster_list$`5`, size=0.5)+
  theme_void()
```



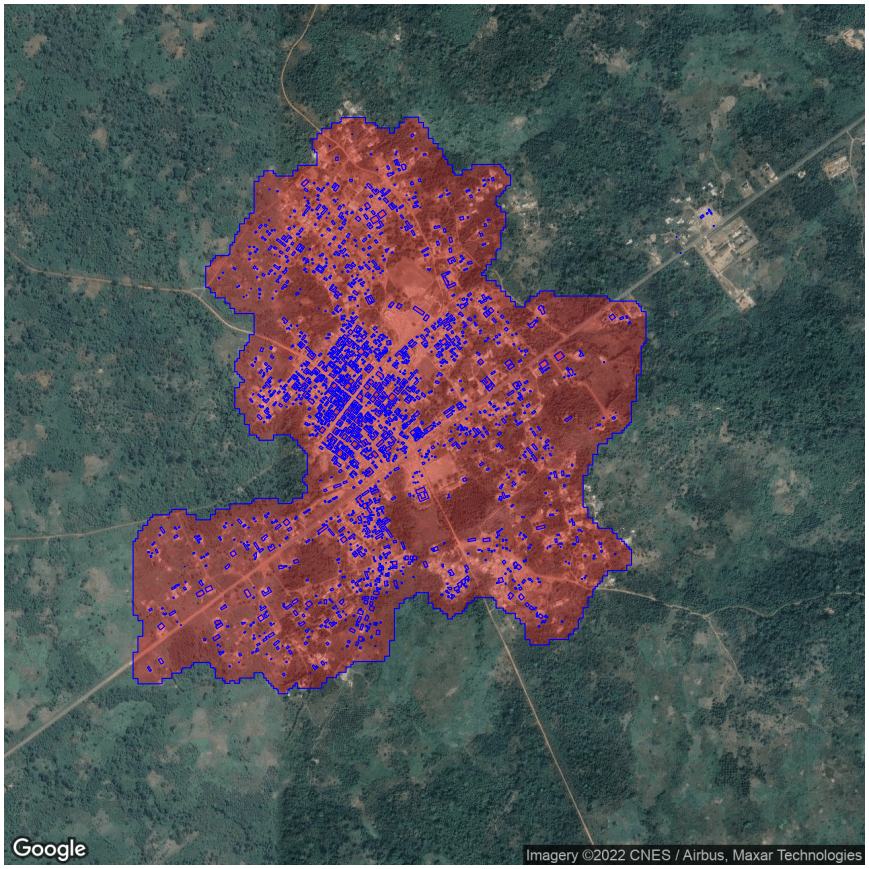
```
#draw polygons around each community cluster
cluster_poly<- foreach(i=1:length(cluster_list)) %do% {

    concaveman(cluster_list[[i]], length_threshold = 0, concavity = 2)


}


#example of polygon around community cluster
#note boundaries intersect buildings on edge of community
ggplot()+
 geom_sf(data = cluster_poly[[5]])+
 geom_sf(data = Example)+
  geom_sf(data = cluster_list$`5`, size=0.5)+
  theme_void()
```

```
#add a 50m buffer around community boundary
cluster_poly_buff<- foreach(i=1:length(cluster_poly)) %do% {

  a<- st_make_valid(cluster_poly[[i]])
  b<- st_buffer(a, dist = 50)
  c<- nngeo::st_remove_holes(b)

  cbind(clusterID=names(cluster_list)[[i]], c)
```

```
}



#boundary now encompasses all buildings in the cluster
ggplot()+
  geom_sf(data = cluster_poly_buff[[5]])+
  geom_sf(data = Example)+
 geom_sf(data = cluster_list$`5`, size=0.5)+
  theme_void()
```

```
ggmap(base)+
coord_sf(crs = st_crs(4326)) +
geom_sf(data = cluster_poly_buff[[5]],
fill=alpha(c("red"), .3),
color="blue",
inherit.aes = FALSE)+
geom_sf(data = Example,
fill=alpha(c("red"), .3),
color="blue",
inherit.aes = FALSE)+
theme(axis.line = element_blank(),
axis.text = element_blank(),
axis.ticks = element_blank(),
plot.margin = unit(c(0, 0, -1, -1), 'lines')) +
xlab('') +
```

```
ylab('')+
scale_fill_manual(values = alpha(c("red"), .3))
```



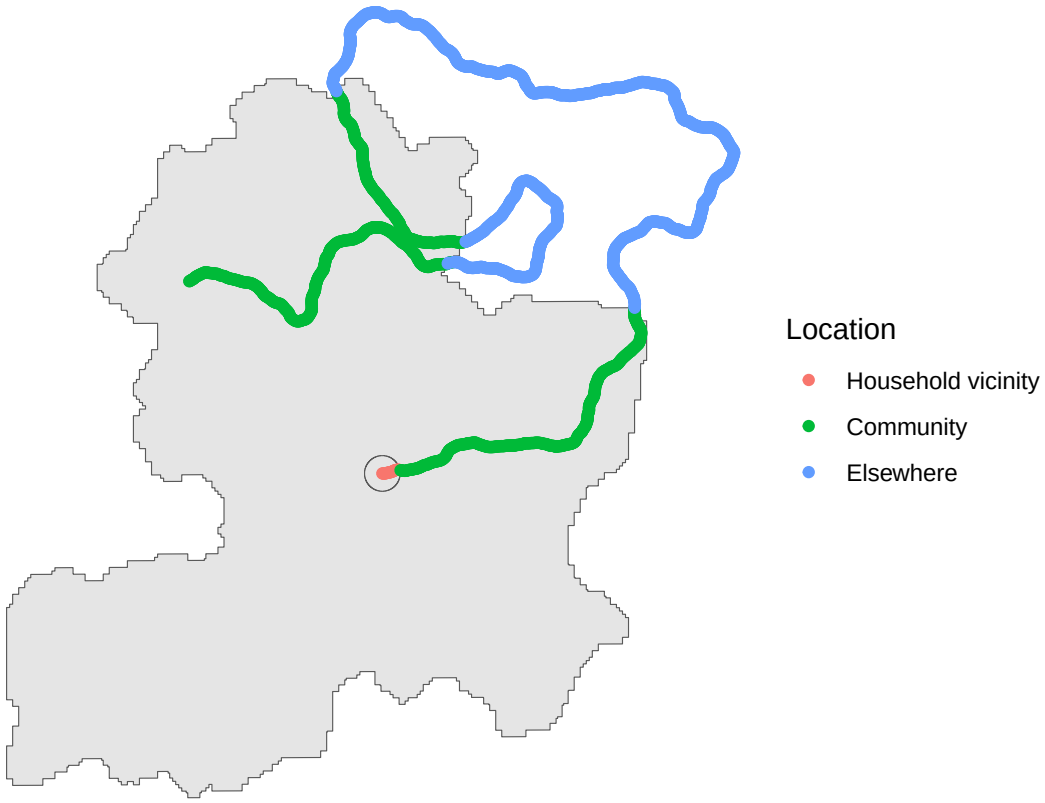

```
#bind list to sf dataframe
community_clusters<- bind_rows(cluster_poly_buff)


#to view all communities with an interactive map
#google_map(key = map_key) %>%
 #    add_polygons(data =Example, polyline = "geometry") %>%
  #add_polygons(data =community_clusters, polyline = "geometry", mouse_over = "clusterID",
#fill_opacity =0, stroke_colour = "#FFFFFF",stroke_opacity = 1, stroke_weight = 3)
```

Example GPS trajectory

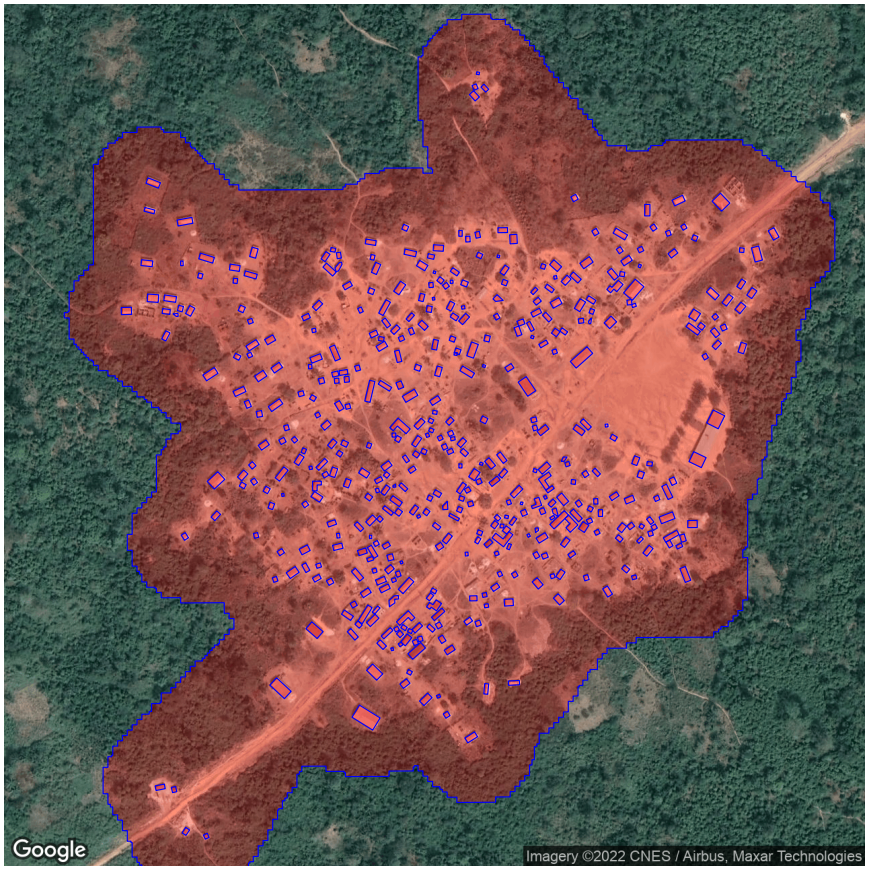
Generate a simulated trajectory and classify points that are in the household vicinity (50m buffer), community boundaries and elsewhere

```
ExampleCentriod<- data.frame(x=-2.446475,y=6.863755)
ExampleCentriod_sf<- st_as_sf(ExampleCentriod, coords=c("x","y"),  crs= 4326)

# Create a random trajectory
set.seed(123)
trj <- TrajGenerate(n = 2880, stepLength = 2, angularErrorSd = .12)

#for illustration, use the example community centroid as the home coordinates
```

```r
trj<- trj %>% mutate(x=-2.446475 + (x / 6378000) * (180 / pi) / cos(-2.446475 * pi/180),
                     y= 6.863755  + (y / 6378000) * (180 / pi))


trj_sf <- st_as_sf(trj, coords = c("x", "y"),
                crs = 4326)

#create 50m buffer around home coordinates
Home_buff<- st_buffer(ExampleCentriod_sf, 50)

#TRUE/FALSE whether trajectory points intersect home buffer
HH<- lengths(st_intersects(trj_sf, Home_buff))>0
#TRUE/FALSE whether trajectory points intersect community boundaries
Comm<- lengths(st_intersects(trj_sf, cluster_poly_buff[[5]]))>0

#Create location variable
trj_sf<- trj_sf %>% mutate(Household= HH, Community=Comm) %>%
  mutate(Location= ifelse(Household==TRUE, "Household vicinity",
                   ifelse(Community==TRUE & Household==FALSE, "Community", "Elsewhere"))) %>%
  mutate(Location= factor(Location, levels = c("Household vicinity", "Community", "Elsewhere")))


ggplot()+
  geom_sf(data = cluster_poly_buff[[5]])+
    geom_sf(data = Home_buff)+
  geom_sf(data = trj_sf, aes(color=Location))+
  theme_void()+
  scale_color_manual(values = rev(c("#619CFF", "#00BA38", "#F8766D")))
```




```
Example<- BuildingsBound_sf %>%
  filter(between(latitud, (6.943246-0.01), (6.943246+0.01))) %>%
  filter(between(longitd,  (-2.515139-0.01), (-2.515139+0.01)))

base<- get_map(location = c(-2.515139, 6.943246), zoom=17, maptype = "satellite")


ggmap(base)+
coord_sf(crs = st_crs(4326)) +
geom_sf(data = cluster_poly_buff[[42]],
fill=alpha(c("red"), .3),
color="blue",
inherit.aes = FALSE)+
geom_sf(data = Example,
fill=alpha(c("red"), .3),
color="blue",
inherit.aes = FALSE)+
theme(axis.line = element_blank(),
axis.text = element_blank(),
axis.ticks = element_blank(),
plot.margin = unit(c(0, 0, -1, -1), 'lines')) +
xlab('') +
ylab('')+
scale_fill_manual(values = alpha(c("red"), .3))
```


Google
Imagery ©2022 CNES / Airbus, Maxar Technologies