Project Four Presentation

Spatial Analytics

Paul M. Cleary

CE 597, Geospatial Data Analytics

Professor: Dr. Jie Shan

October 16, 2019

BURKE GRADUATE PROGRAM; LYLES SCHOOL OF CIVIL ENGINEERING

PURDUE UNIVERSITY

Agenda

- Data Cleaning
- Functions (Perimeter, Area, Centroid)
  - My functions vs. R functions
- Spatial Join
  - Number of tweets per building
- Conclusion
  - Difficulties

Data Cleaning

Dealing with Invalid Geometries and Demolished Buildings

```
setwd("~/Fall 19/Data Analytics/Project4")
# Read in the shapefile of tract data
dataimport <- st_read('buildings_shpfile/buildings.shp')
dataimport <- subset(dataimport, is.na(YEAR_RAZED))
campus_bldgs <- dplyr::select(dataimport,Entity,BLDG_ABBR,BUILDING_N,GIS_BID,Shape_Leng,Shape_Area,geometry)
# Project the data into UTM Zone 16N, WGS 84
campus_bldgs <- st_transform(campus_bldgs,32616)
# Check for empty geometries
check <- any(is.na(st_dimension(campus_bldgs$geometry)))
# Check for corrupt geometries
check2 <- any(is.na(st_is_valid(campus_bldgs$geometry)))
# Check for invalid geometries
check3 <- any(na.omit(st_is_valid(campus_bldgs$geometry)) == FALSE)
# I have empty, corrupt, and invalid geometries; use st_make_valid to correct them.
#Single polygons may become multi-geometries in case of self-intersections.
# Remove anything with empty geometries.
campus_bldgs <- campus_bldgs[!st_is_empty(campus_bldgs),]
campus_bldgs <- st_make_valid(campus_bldgs)
campus_bldgs <- st_collection_extract(campus_bldgs, "POLYGON") # Extract MultiPolygons from Geometry Collections
campus_bldgs <- st_cast(campus_bldgs, to = "POLYGON")
```

Data Cleaning

https://www.r-spatial.org/r/2017/03/19/invalid.html

Tidying feature geometries

When you analyse your spatial data with `sf` and you don't get any warnings or error messages, all may be fine. In case you do, or your are curious, you can check for

1. empty geometries, using `any(is.na(st_dimension(x)))`
2. corrupt geometries, using `any(is.na(st_is_valid(x)))`
3. invalid geometries, using `any(na.omit(st_is_valid(x)) == FALSE)`; in case of corrupt and/or invalid geometries,
4. in case of invalid geometries, query the reason for invalidity by `st_is_valid(x, reason = TRUE)`
5. you may be succesful in making geometries valid using `st_make_valid(x)` or, if `st_make_valid` is not supported by
6. `st_buffer(x, 0.0)` on non-corrupt geometries (but beware of the bowtie example above, where `st_buffer` removes one half).
7. After succesful a `st_make_valid`, you may want to select a particular type subset using `st_is`, or cast `GEOMETRYCOLLECTIONS` to `MULTIPOLYGON` by

Functions

My functions for the perimeter, area, and centroid of a polygon

```r
# create a function to find the perimeter of each p
my_perimeter <- function(d){
  m <- as.matrix(d[[1]])
  x <- m[ ,1]
  y <- m[ ,2]
  vertices <- length(m)
  i <- 1
  j <- i + 1
  perimeter = 0
  for (val in x){
    len <- sqrt((y[j] - y[i])^2 + (x[j] - x[i])^2)
    j <- j + 1
    i <- i + 1
    perimeter = perimeter + len
    if(i == vertices/2) {j == 1}
    if (i == vertices/2)
      break
  }
  return(perimeter)
}
```

```r
# Function to find the area of each polygon
my_area <- function(d){
  m <- as.matrix(d[[1]])
  x <- m[ ,1]
  y <- m[ ,2]
  vertices <- length(m)
  i <- 1
  j <- (vertices/2)
  area = 0
  for (val in x){
    area <- area + (x[j] - x[i])*(y[i] + y[j])
    j <- i
    i <- i + 1
    }
area <- (area/2) * -1
return(area)
}
```

```r
my_centroid <- function(d,A){
  m <- as.matrix(d[[1]])
  x <- m[ ,1]
  y <- m[ ,2]
  temp <- nrow(m) -1
  m <- m[1:temp, ]
  x_cent <- 0
  y_cent <- 0
  for (i in 1:nrow(m)){
    x_cent[i] <- -(y[i+1] - y[i])*(x[i]^2 + (x[i]*x[i+1]) + x[i+1]^2)
    y_cent[i] <- -(x[i] - x[i+1])*(y[i]^2 + (y[i]*y[i+1]) + y[i+1]^2)
  }
  x <- sum(x_cent)/(A*6)
  y <- sum(y_cent)/(A*6)
  return(c(x,y))
}
```

Functions

R Built in Functions

```
 98  # Create Columns of perimeter, area, and centroid with r functions
 99  campus_bldgs["r_perimeter"] <- st_perimeter(campus_bldgs$geometry)
100  campus_bldgs["r_perimeter"] <- as.numeric(round(campus_bldgs$r_perimeter, digits = 3))
101  campus_bldgs["r_area"] <- st_area(campus_bldgs$geometry)
102  campus_bldgs["r_area"] <- as.numeric(round(campus_bldgs$r_area, digits = 3))
103  r_cent <- mapply(st_centroid,campus_bldgs$geometry)
104  r_cent <- t(r_cent)
105  r_centdf <- data.frame(r_cent)
106  r_centdf <- round(r_centdf, digits = 3)
107  r_centdf <- unite(r_centdf, newcol, c(X1, X2), remove=FALSE)
108  campus_bldgs["r_centroid"] <- r_centdf$newcol
```

Functions

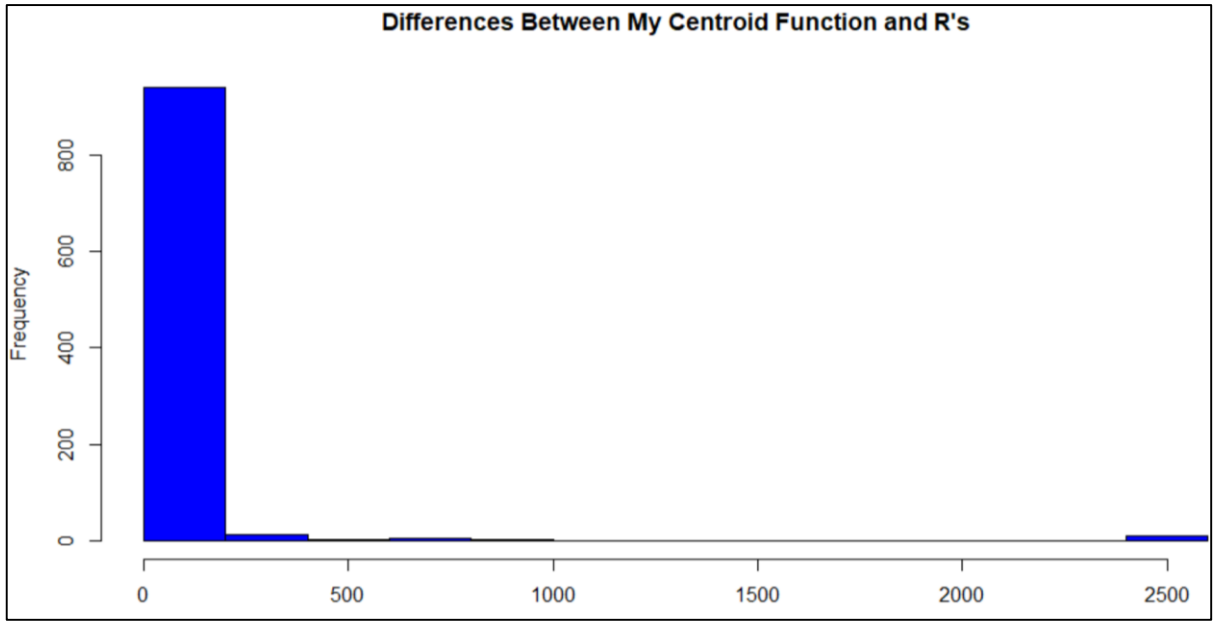
Function results

my_perimeter	my_area	my_centroid	r_perimeter	r_area	r_centroid	diff_perimeters	diff_areas	diff_centroids
180.477	1822.598	506859.48_4473866.969	180.477	1822.598	506859.549_4473867.583	0	0	0.618
23.040	41.754	498766.079_4482734.452	23.040	41.754	498765.183_4482726.397	0	0	8.104
23.123	42.076	498772.773_4482748.998	23.123	42.076	498769.587_4482720.365	0	0	28.811
23.136	42.145	498770.646_4482717.41	23.136	42.145	498770.122_4482712.705	0	0	4.734
23.057	41.895	498762.438_4482677.595	23.057	41.895	498765.583_4482705.858	0	0	28.437
77.925	482.229	498343.156_4483010.705	77.925	482.229	498343.589_4483014.607	0	0	3.926
13.934	15.060	498368.806_4483112.979	13.934	15.060	498354.526_4482984.525	0	0	129.245
223.138	2895.374	499158.407_4482861.102	223.138	2895.374	499158.481_4482861.765	0	0	0.667
180.700	1134.531	499158.111_4482833.208	180.700	1134.531	499157.942_4482831.692	0	0	1.525
324.060	7017.949	499158.373_4482774.379	324.060	7017.949	499158.388_4482774.518	0	0	0.140
232.256	3409.274	499054.365_4483164.072	232.256	3409.274	499054.424_4483164.605	0	0	0.537
174.281	1217.436	499054.723_4483128.725	174.281	1217.436	499054.741_4483128.886	0	0	0.163

Functions

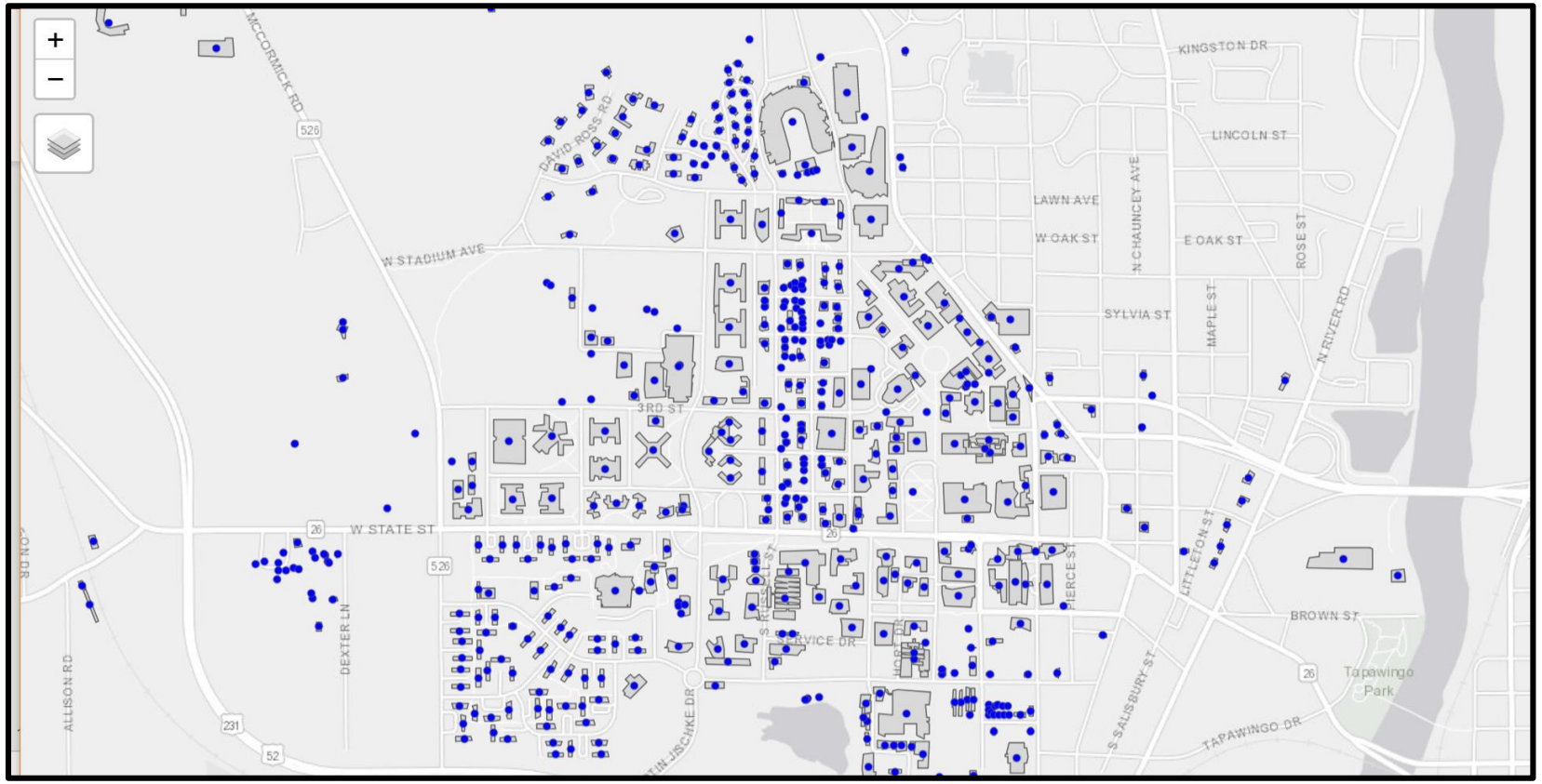
Statistics Between the Functions

Statistics	Perimeter Diff	Area Diff	Centroid Diff
Max	0	0	2475.177 m
Mean	0	0	41.155 m
SD	0	0	245.385 m
Min	0	0	0.001 m



Functions

Buildings with R function Centroids



BURKE GRADUATE PROGRAM; LYLES SCHOOL OF CIVIL ENGINEERING

PURDUE UNIVERSITY

Spatial Join

Joining Building Polygons and Tweet Locations

```
#Read in twitter data
Tweets2014 <- read.csv("pu2014.csv")
Tweets <- subset(Tweets2014, select = c("epoch","user_id","longitude","latitude"))
Tweets <- st_as_sf(Tweets, coords = c("longitude", "latitude"))
st_crs(Tweets) <- 4326
Tweets <- st_transform(Tweets, 32616)

# Join Twitter data and Campus polygons
bldgs_tweets <- st_join(Tweets, campus_bldgs, join = st_within)
# Count the # of tweets per building
t_per_bldg <- count(as_tibble(bldgs_tweets), BUILDING_N)
# Show tweets with cleaned up building polygons
tmap_mode("view")
campus_map <- tm_shape(campus_bldgs) + tm_polygons()
  + tm_shape(bldgs_tweets) + tm_dots(col = "blue")
print(campus_map)
```

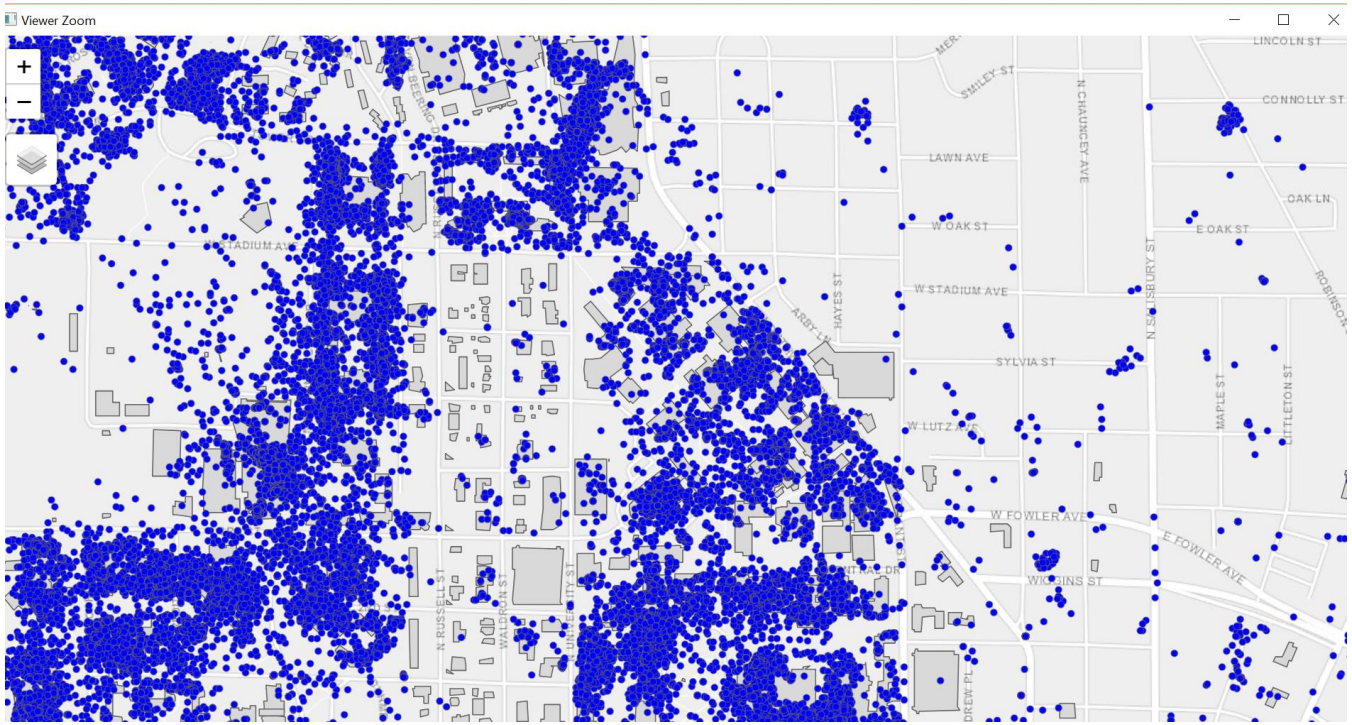
	BUILDING_N	n
233	*NA*	48328
232	Cordova Recreational Sports Center	1925
231	Richard Owen Residence Hall	1701
230	John T. Mccutcheon Residence Hall	1213
229	Virginia C. Meredith Residence Hall	910
228	Harvey W. Wiley Residence Hall	840
227	Richard Benbridge Wetherill Lab Of Chemistry	688
226	John W. Hicks Undergraduate Library	660
225	Edward C. Elliott Hall Of Music	607
224	Newton Booth Tarkington Residence Hall	601
223	Benjamin Harrison Residence Hall	581

Conclusion

Challenges

Sfc objects

- Breaking apart for centroid distance calculations
- Creating sfc multipoint to plot centroids created with my function



PURDUE UNIVERSITY