Zoo955 - Week 3

Hilary Dugan

2/13/2018

Question 1

Define functions from the `sf` package. Good resource: https://en.wikipedia.org/wiki/DE-9IM

`st_intersects` = For a pair of objects (each object containing one or more points, lines, or polygons), st_intersects tells you whether they intersect. "Intersect" means two geometries share any portion of space, and functions such as "overlaps" and "touches" are subcategories of intersection. Returns intersecting geometries as either a list (sparse = TRUE) or a matrix (sparse = FALSE). Some good documentation here: https://cran.r-project.org/web/packages/sf/vignettes/sf1.html#geometrycollection

`st_disjoint` = This is the opposite of st_intersects. Disjointed objects do not share any space together.

`st_touches` = point/line, point/poly, line/poly –must be on end point; poly/poly–must not share internal space

`st_crosses` = only allowable with line/line and line/poly (not poly/poly). crossing does not include endpoints ever, and lines must exist inside and outside of polygons.

`st_within` = True if A is completely inside of B

`st_contains` = True if no point in B lies in the exterior of A, and at least one point lies inside of A

`st_overlaps` = if the shapes share space, i.e., intersect, but one is not completely within another, will return list of IDs (row number), describing which IDs overlap with one another.

`st_equals` = if the shapes are the same, i.e., occupying the same space, will return list of IDs (row number) describing which IDs are equal to one another.

`st_covers` = This function tests if the geometry of object x covers the geometry of object y. If no points in object y are beyond the boundaries of object x, then object x completely covers object y. https://postgis.net/docs/ST_Covers.html

`st_covered_by` = This is essentially the inverse of st_covers. st_covered_by tests if the geometry of object x is covered by the geomertry of object y. If no points in object x are beyond the boundaries of object y, then object x is completely covered by y (and y completely covers x). st_covers(x,y) will produce the same result as st_covered_by(y,x).

`st_equals_exact` =

`st_is_within_distance` =

`st_buffer` = create a new polygon that all coordinates are X distance away from the edge of another polygon, line or point. The buffer can extend outward (+) or inward (-) of a polygon.

`st_boundary` = this returns the geometry, which is the combinatorial boundary of the given geometric object. For polygons, this is simply the outline. For points, it seems to connect points in the order (example) without closing them

`st_convexhull` = returns the smallest geometry that cantains all geometries for a given shape. It's like putting a rubber band around a shape.

`st_union_cascaded` = Doesn't exist anymore. _cascaded refered to a new, at the time, algorithm that sped up st_union

`st_simplify` =

`st_triangulate` = connects 3D cloud of points with triangles to create a connected surface called a TIN, it's often used in analysis of topography

`st_polygonize` = creates a multipolygon feature from lines of an input geometry. If the input is lines, it will create polygons only if enclosed by overlapping vertices.

`st_centroid` =

`st_segmentize` =

`st_union` = returns a polygon or line feature where polygons/lines are combined if they overlap/touch

Question 2

Make a 500 m buffer of the 4 southern LTER lakes. Which buffers overlap?

Lakes Data: https://lter.limnology.wisc.edu/dataset/north-temperate-lakes-lter-yahara-lakes-district-boundary

Read in shapefiles using `readOGR`

```
library(sf)
lakes = st_read(dsn = '../Lecture3_Shapefiles/Data/yld_study_lakes.shp',stringsAsFactors = F)
```

```
## Reading layer `yld_study_lakes' from data source `C:\Users\hdugan\Documents\Rpackages\Zoo955\Lecture3
## Simple feature collection with 4 features and 9 fields
## geometry type:  POLYGON
## dimension:      XY
## bbox:           xmin: 547589.4 ymin: 286020.8 xmax: 574950 ymax: 313254
## epsg (SRID):    NA
## proj4string:    +proj=tmerc +lat_0=0 +lon_0=-90 +k=0.9996 +x_0=520000 +y_0=-4480000 +ellps=GRS80 +tow
```

```
# What are the lake IDs?
lakes$LAKEID
```

```
## [1] "FI" "ME" "MO" "WI"
```

Make 500 m buffer

```
buffer500 = st_buffer(lakes,500)
#Check if buffers overlap
st_overlaps(buffer500)
```

```
## Sparse geometry binary predicate list of length 4, where the predicate was `overlaps'
##  1: (empty)
##  2: 3
##  3: 2, 4
##  4: 3
```

We know the order of the lake IDS (FI, ME, MO, WI). So based on the overlap matrix.

- Mendota overlaps with Monona.
- Monona overlaps with Mendota and Wingra
- Wingra overlaps with Monona

Question 3

Increase the size of the lakes by 2x. What is the percent of Mendota that overlaps with Monona?

Note:

- You can't use a buffer because that does not retain the shape of the lakes.
- You can't just multiply the lakes x2, because that multiplies the coordinates. You end up with the lakes somewhere other than Wisconsin.

Instead:

- Find the distance from the edge of the lake to the centroid. Multiply these distances by 2.

```
# Take just the geometry of the lakes
glakes = st_geometry(lakes)
# Find the centroid
cntrd = st_centroid(glakes)

# Find distance from edge of lakes to centroid
cDist = (glakes - cntrd)
# Multiply this distance by 2 and add back to centroid
glakes2 = cDist * 2 + cntrd

# Find the intersection between Mendota and Monona. We know these are lakes 2 and 3.
int = st_intersection(glakes2[2],glakes2[3])
# What is the size difference between the intersection and Mendota
st_area(int)/ st_area(glakes2[2])
```

```
## [1] 0.1942473
```

The percent of Mendota that overlaps with Monona is 19.4%

If you want to double check by plotting

```
# Plot glakes and centroid
plot(glakes,col='cadetblue')
plot(cntrd,col='red3',pch=16,add=T)
# Plot lakes double the size
plot(glakes2,border='red4',add=T)
# Plot intersection
plot(int,col='red4',add=T)
```