Loken_HW3

Hilary Dugan + Luke Loken

February 12, 2018

Hilary's answers

Question 1 Define functions from the `sf` package. Good resource: https://en.wikipedia.org/wiki/DE-9IM

`st_intersects` =

`st_disjoint` =

`st_touches` =

`st_crosses` =

`st_within` =

`st_contains` =

`st_overlaps` =

`st_equals` =

`st_covers` =

`st_covered_by` =

`st_equals_exact` =

`st_is_within_distance` =

`st_buffer` =

`st_boundary` =

`st_convexhull` =

`st_union_cascaded` =

`st_simplify` =

`st_triangulate` =

`st_polygonize` =

`st_centroid` =

`st_segmentize` =

`st_union` =

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

```
library(sp)
library(rgdal)
library(raster)
library(sf)
```

```
library(tidyr)
library(dplyr)
Lakes_sp <- readOGR(paste0("E:/Git_Repo/ZOO955/Data/Shapefiles"), "yld_study_lakes", stringsAsFactors =
```

```
## OGR data source with driver: ESRI Shapefile
## Source: "E:/Git_Repo/ZOO955/Data/Shapefiles", layer: "yld_study_lakes"
## with 4 features
## It has 9 fields
```

```
Lakes_sf <- st_read("E:/Git_Repo/ZOO955/Data/Shapefiles/yld_study_lakes.shp", stringsAsFactors = F)
```

```
## Reading layer `yld_study_lakes' from data source `E:\Git_Repo\ZOO955\Data\Shapefiles\yld_study_lakes
## 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 +tou
```

```
#Make individual lake objects
ME <- Lakes_sf %>% filter(LAKEID == 'ME')
MO <- Lakes_sf %>% filter(LAKEID == 'MO')
WI <- Lakes_sf %>% filter(LAKEID == 'WI')
FI <- Lakes_sf %>% filter(LAKEID == 'FI')

#Transform to lat/long for axes plotting
Lakes_sf_latlong <- st_transform(Lakes_sf, crs=4326)

#Merging Hilary's and Luke's objects
lakes <- Lakes_sf

# What are the lake IDs?
ids <- lakes$LAKEID
names <- lakes$LAKE_NAME
```

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

Or we can add lake ids to table

```
# Repeat with sparse=False
overlaps <- st_overlaps(buffer500, sparse = FALSE)
colnames(overlaps) <- ids
```

```r
rownames(overlaps) <- ids
print(overlaps)
```

```
##       FI    ME    MO    WI
## FI FALSE FALSE FALSE FALSE
## ME FALSE FALSE  TRUE FALSE
## MO FALSE  TRUE FALSE  TRUE
## WI FALSE FALSE  TRUE FALSE
```

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.

```r
# 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
```

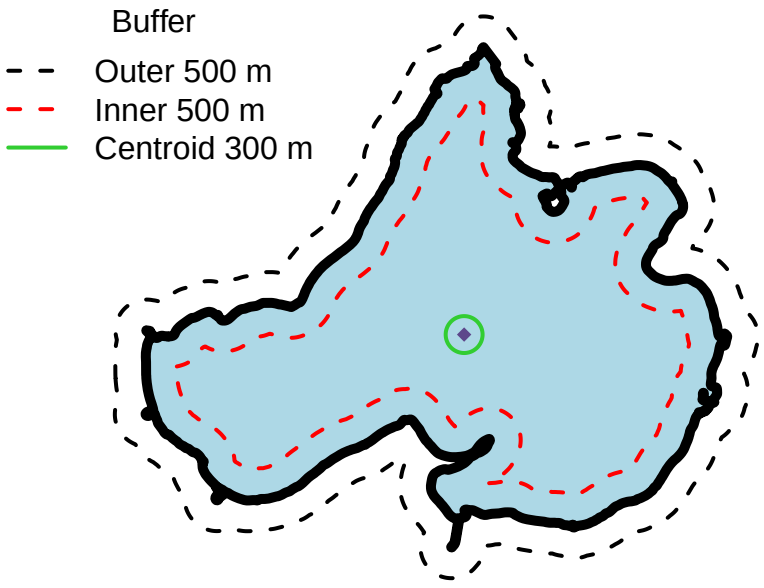
The percent of Mendota that overlaps with Monona is 19.4%

If you want to double check by plotting

```r
# 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)
```

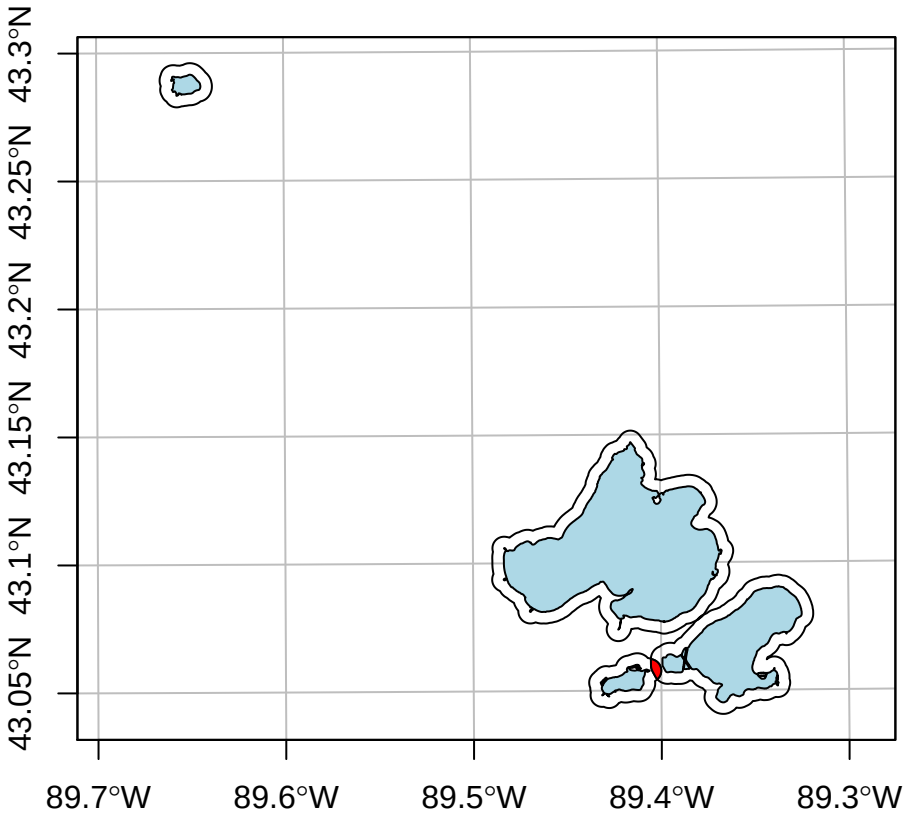
Dear Dr. Dugan:

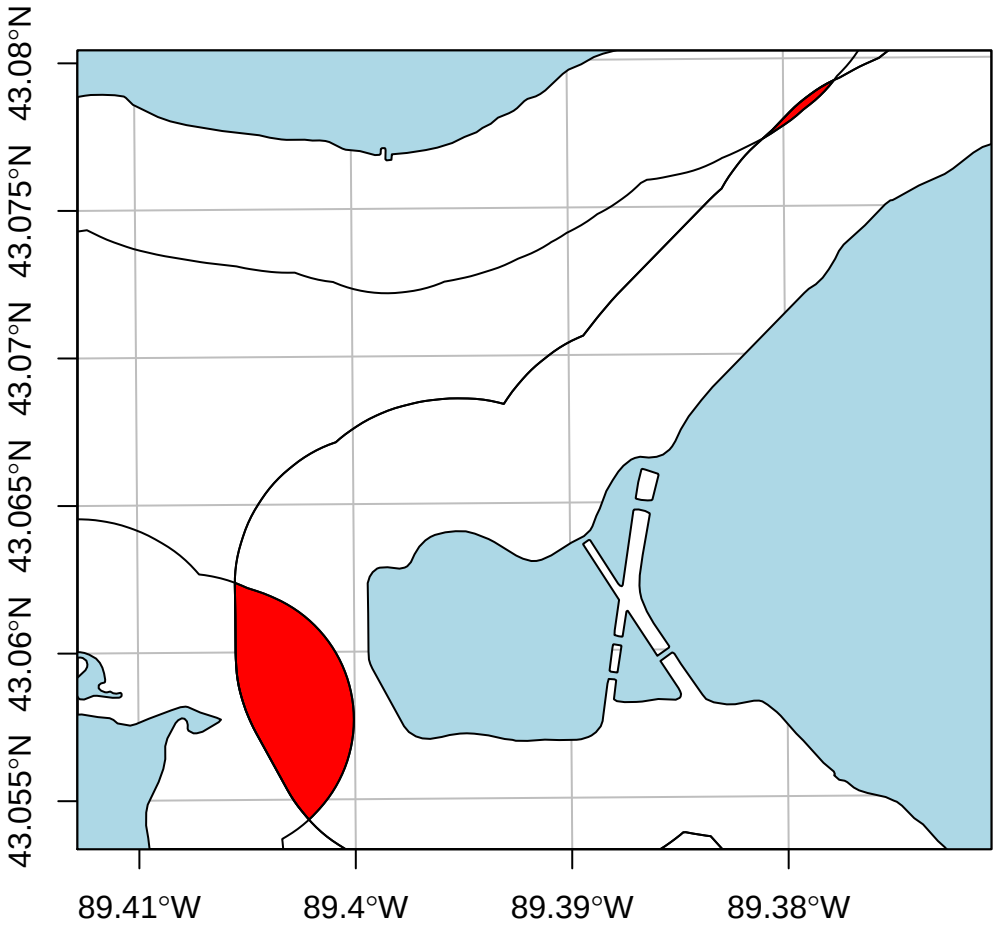
Question 1: What does `st_buffer` do?



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

Question 2: Make a 500 m buffer around the 4 southern LTER lakes, which buffers overlap?





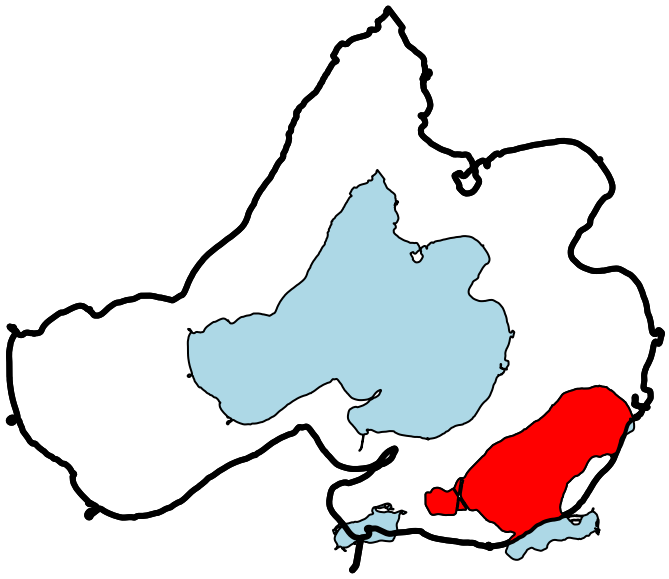
Based on `st_overlaps`, the buffer around Lake Monona (ID#3) overlaps both the Lake Mendota (ID#2) and Lake Wingra (ID#4) buffers. No other polygons overlap.

```
overlaps<-st_overlaps(Lakes_sf500)
names(overlaps)<-Lakes_sf500$LAKE_NAME
str(overlaps)
```

```
## List of 4
##  $ Fish Lake   : int(0)
##  $ Lake Mendota: int 3
##  $ Lake Monona : int [1:2] 2 4
##  $ Lake Wingra : int 3
##  - attr(*, "predicate")= chr "overlaps"
##  - attr(*, "region.id")= chr [1:4] "1" "2" "3" "4"
##  - attr(*, "ncol")= int 4
##  - attr(*, "class")= chr "sgbp"
```

Question 3: Increase the size of the lakes by 2x, What percent of Mendota overlaps with Monona?

```
ME_centroid<-st_centroid(ME)
ME_points<-st_cast(ME, "POINT")

x<-st_coordinates(ME_points)[,1]
y<-st_coordinates(ME_points)[,2]

x2<-2*x-st_coordinates(ME_centroid)[,1]
y2<-2*y-st_coordinates(ME_centroid)[,2]

ME_points2<-st_multipoint(as.matrix(data.frame(x2, y2)))
ME_points3<-st_sfc(ME_points2)

ME_Lines<-st_multilinestring(ME_points3)
ME_Polygon<-st_polygonize(ME_Lines)

ME_Polygon2<-st_geometry(ME_Polygon, type = 3)
st_crs(ME_Polygon2) <- st_crs(ME)
overlap<-st_intersection(ME_Polygon2, MO)
```



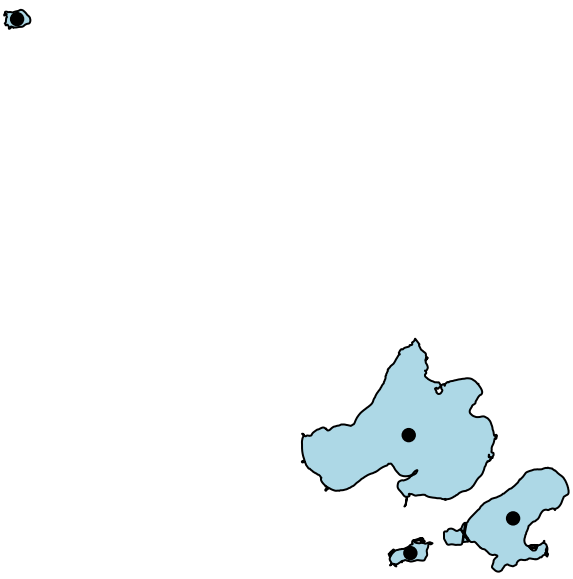
```
overlap_area<-st_area(overlap)
BigMendota_area<-st_area(ME_Polygon2)
as.numeric(overlap_area/BigMendota_area)
```

```
## [1] 0.07484969
```

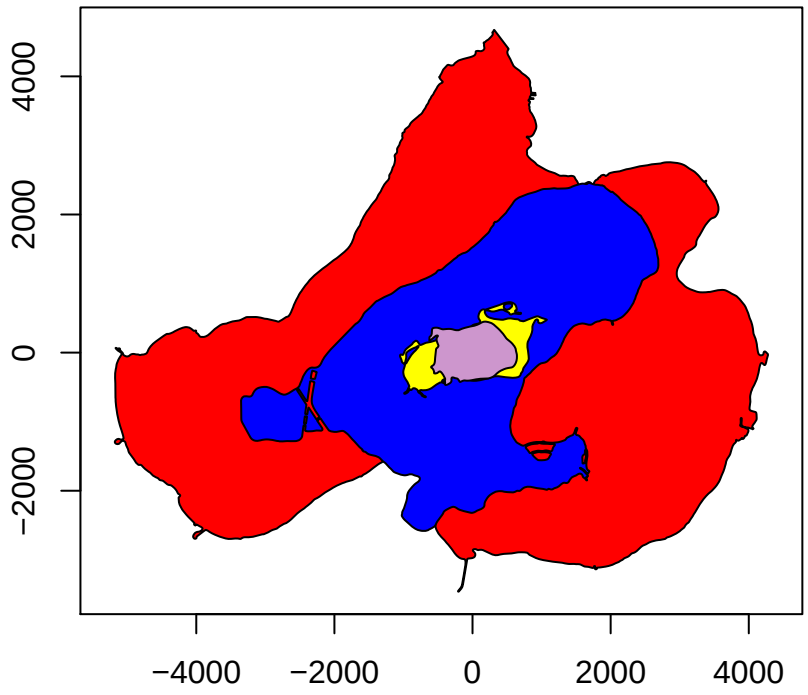
7.45% of the expanded Lake Mendota polygon overlaps Lake Monona.

More code to work through question 3

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

plot(glakes, col='lightblue')
plot(cntrd, add=T, pch=16)
```



```
# Find distance from edge of lakes to centroid
cDist = (glakes - cntrd)
plot(cDist[c(2,3,4,1)], col=c('red', 'blue', 'yellow', 'plum3'), axes=T)
```



Note that `cntrd` is multiple polygon, where all polygons are centered at zero (centroid) and the x/y coordinates are distances **i.e., not coordinates**

```
# 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])
```

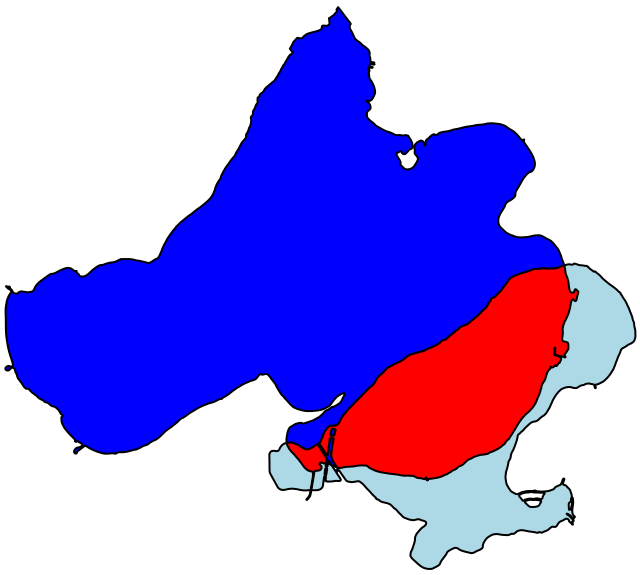
```
## [1] 0.1942473
```

Another way to do it, without creating a 'distance' polygon

```
# using geometry you can simply double the coordinates and subtract by the centroid
glakes3 = glakes*2 - cntrd

int = st_intersection(glakes3[2],glakes3[3])

st_area(int)/ st_area(glakes3[2])
```

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

```
plot(glakes3[2:3], col=c('blue', 'lightblue'))
plot(int, col=c('red'), add=T)
```



This does not work if you use the sf object. This code creates an error `lakes3 <- lakes*2 - st_centroid(lakes)`

But if you use the `st_geometry` function you can do geometry math

```
biglakes <- st_geometry(lakes)*2 - st_centroid(st_geometry(lakes))
plot(biglakes, col=c('red', 'blue', 'yellow', 'plum3'))
plot(lakes, add=T, col=NA)
```