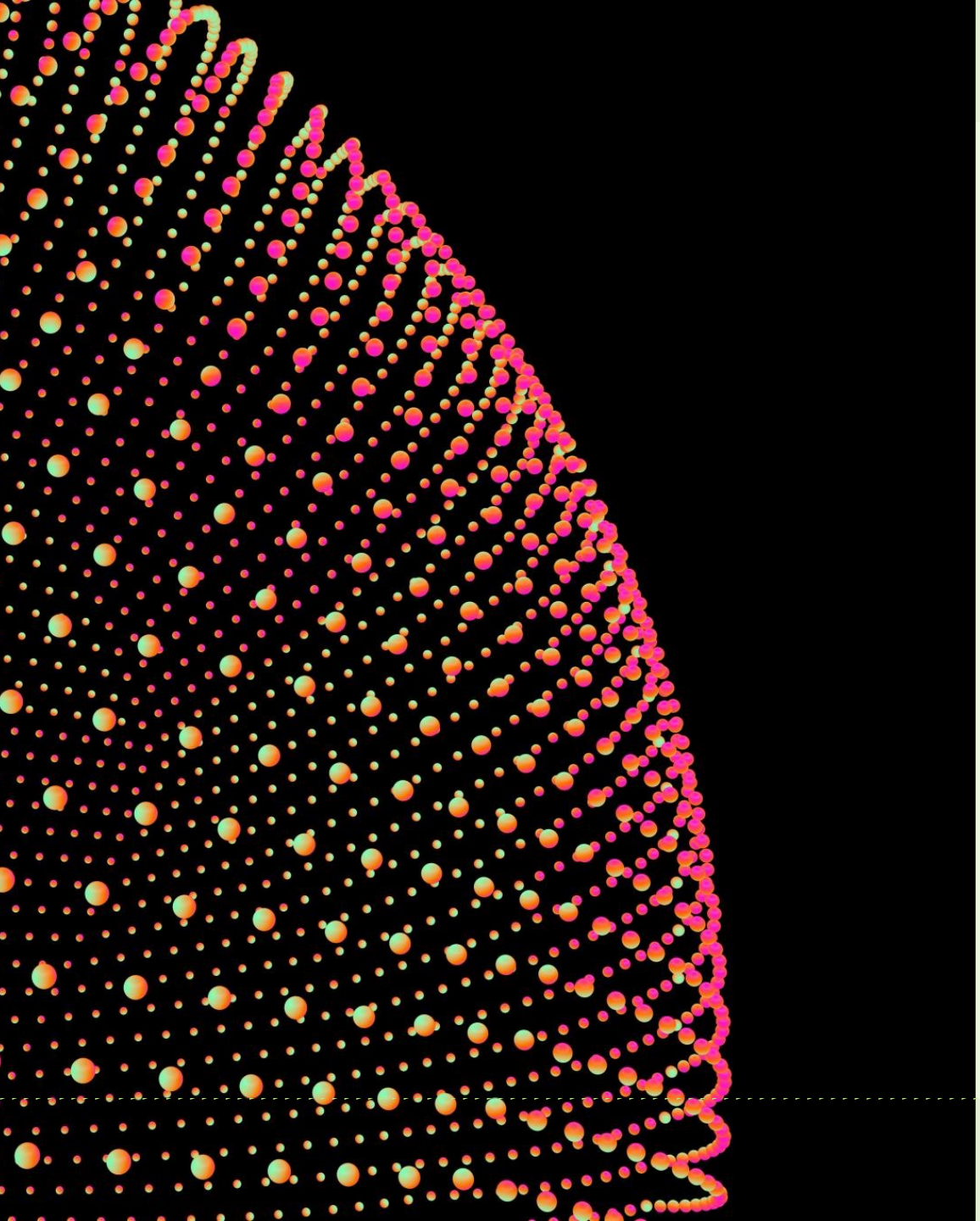


ECON6027 2a

Geometry Operations (on Vector Data)

Packages you need

sf

spData

Geometry operations on vector data

Chapter 1 focused on handling attribute and spatial elements of an sf object.

This lesson is on how to handle the geometry column of the sf object.

We will use the seine dataset (for the most part).

```
> class(seine)
> seine
```

SMU Classification: Restricted

Simplify

If the dataset is too large (i.e. the features are overly detailed), then in order to save RAM and increase computational speed, we may need to simplify the geometry.

```
> seine_simp = st_simplify(seine)
```

```
> object.size(seine)
```

18096 bytes

```
> object.size(seine_simp)
```

17912 bytes

"dTolerance" can be used to control the level of generalization in map units. This argument can be used to increase/decrease the level of simplification. E.g. `> seine_simp = st_simplify(seine, dTolerance = 2000)` sets a generalization of up to 2km. (Use `> st_crs(seine)$units_gdal` to check the map units.)

Exercise A

st_simplify can be applied on polygons as well but may cause topology to be lost. Check this for yourself using “nz” dataset. (Set the tolerance to 20,000 square meter)

Centroids: identify center of spatial objects

Calculates the center of mass of a spatial object

```
> seine_cen = st_centroid(seine)
```

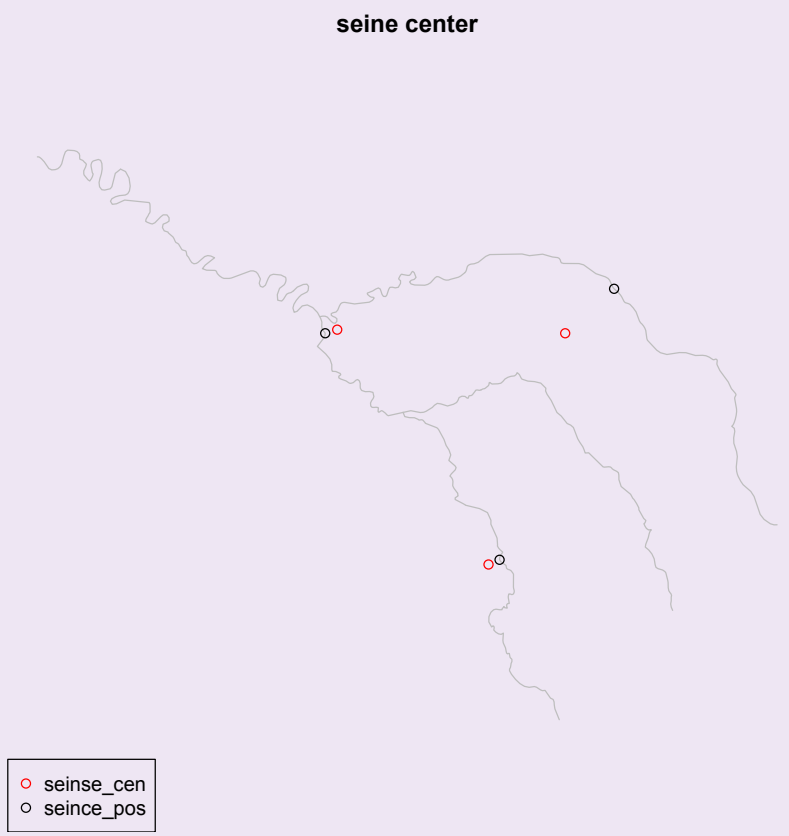
To ensure the points are inside the geometry of the parent object

```
> seine_pos = st_point_on_surface(seine)
```

```
> plot(seine, main="seine center", reset=F,
col="grey"); plot(seine_pos, add=T,
col="black"); plot(seine_cen, add=T, col="red")
```

```
> legend("bottomleft",
c("seinse_cen","seince_pos"),
col=c("red","black"), pch=c(1,1))
```

The two functions assumes attributes are constant over geometries of x. This means the surface is assumed to be flat without indents or bumps.



Exercise B

Plot the center of the regions of NZ “inside” each parent object.

Create Distance Attributes

Distance between two locations

```
>
st_distance(seine_cen
[1,], seine_cen[2,])
```

Compute Euclidian or great circle distance between pairs of geometries (the centroids we captures earlier in this case); compute, the area or the length of a set of geometries.

Buffer

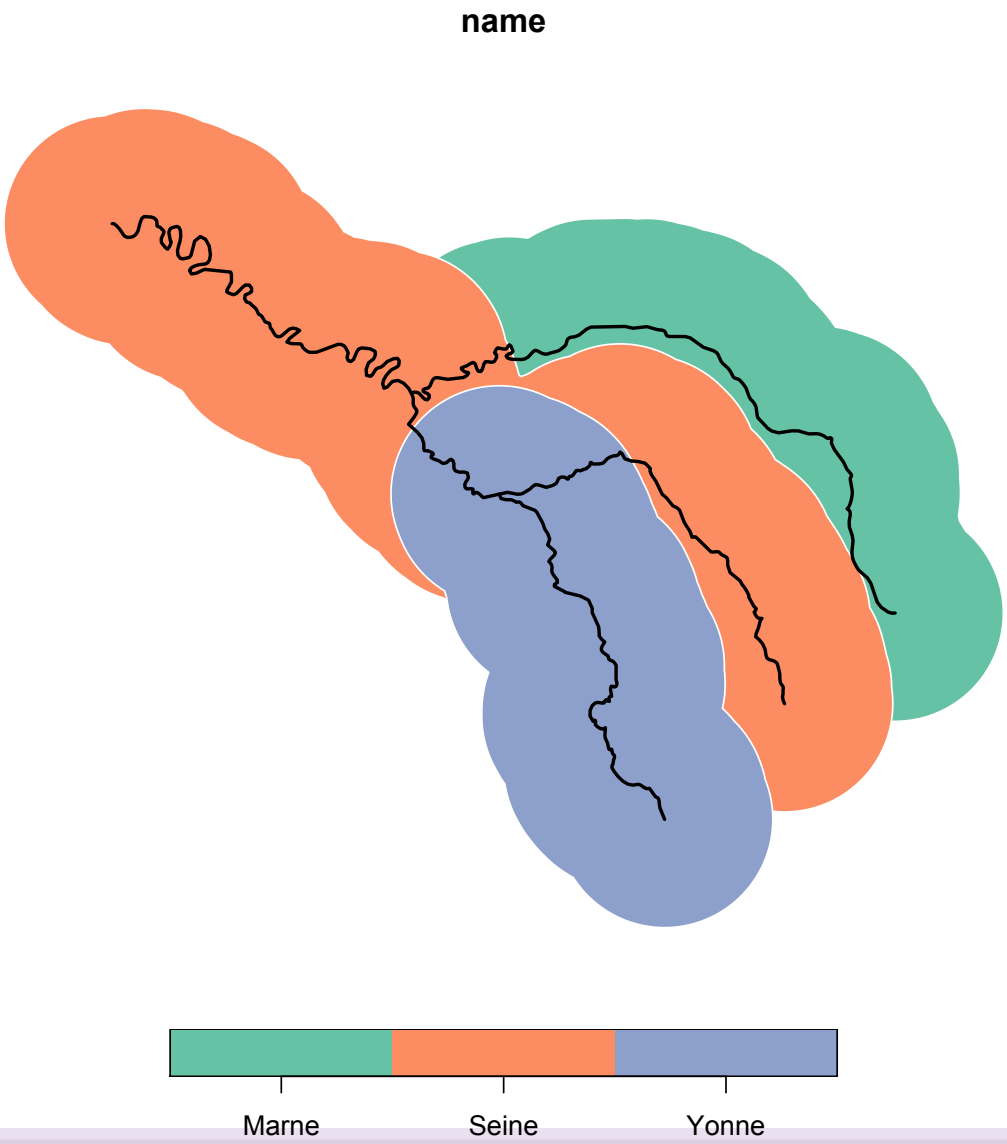
A buffer creates a polygon of a given distance around a geometric feature (point, line or polygon)

Buffers can be useful to answer questions like,

How many **points are within a given distance** from a line/polygon boundary?	Polygons may not intersect per say, but is there a significant **common region** between two polygons?	How **close are points to one another**? (we will take this again when we talk about point pattern analysis)

Create Buffer

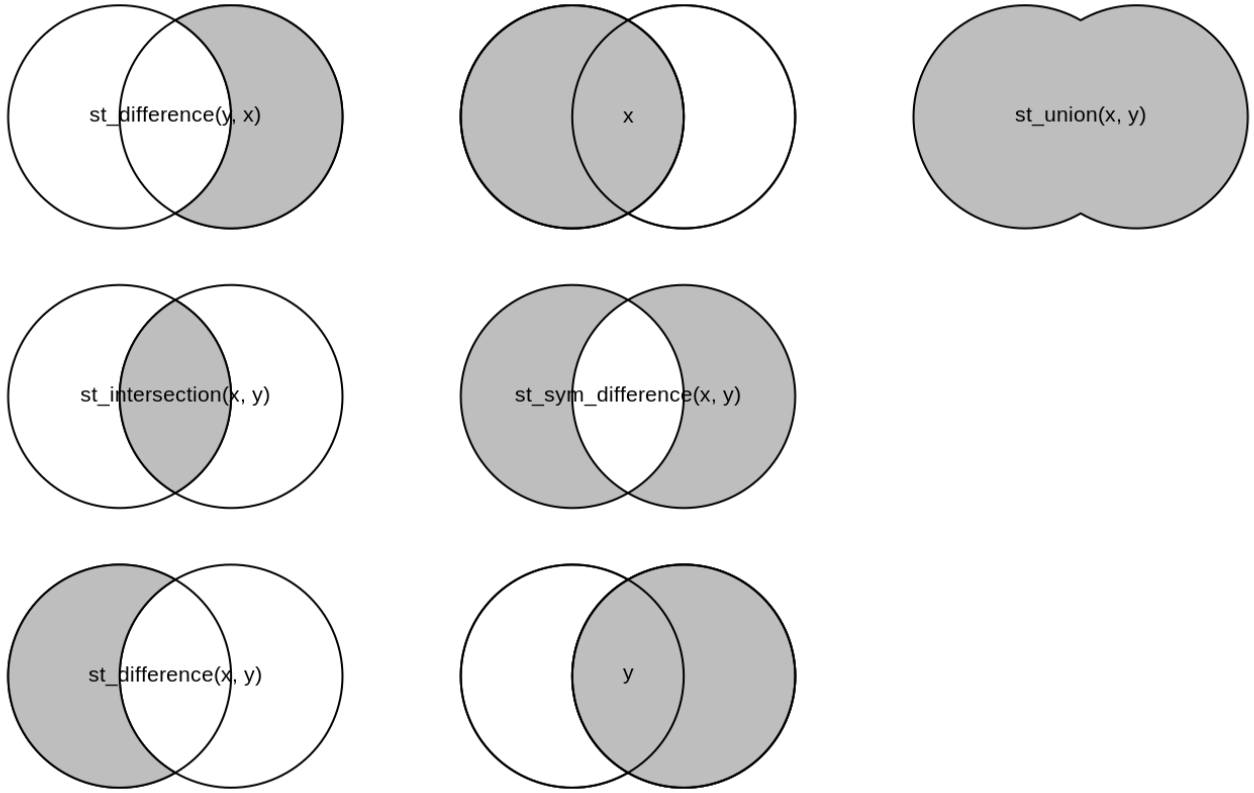
```
> st_length(seine) # check unit

> seine_buff = st_buffer(seine,
dist = 50000) # 50km buffer

> plot(seine_buff,
border="white", reset=F)

> plot(seine, col="black",
lwd=2,add=T)
```



Clip operations

Can you explain what each of the operations mean?

For example the first function defines the region in Y that is not in X, {Y\X}.



SMU Classification: Restricted

Clip operations: Geometry Intersections

```
> seine_buff
```

Subset the buffers

```
> x = seine_buff[1,];  y = seine_buff[2,]
```

Identify the intersecting region between Marne and Seine buffers

```
> xANDy = st_intersection(x, y); plot(xANDy["name"])
```

Clip operations assumes to be spatially constant throughout all geometries.

Exercise C

1. Repeat the above steps to create **intersections** between
   - Marne and Yonne
   - Seine and Yonne
2. Create a **union** of all three buffers.
3. Exlain the difference between st_combine and st_union. Hint: > seine_combine = st_combine(seine_buff)
4. Isolate the buffer of Marne that is not within the buffer of Yonne.

Exercise D

Include the territorial waters to “nz” polygon in a plot (see https://simple.wikipedia.org/wiki/Territorial_waters)

SMU Classification: Restricted

Take home points…

- Manipulate the geometry column of an sf object
  - Simplify
  - Find centroids
  - Compute distances
  - Create buffers
  - Clipping

Important R functions

- st_simplify()
- st_centroid()
- st_point_on_surface()
- st_distance()
- st_length()
- st_buffer()
- st_intersection(), st_union(), st_difference(), etc.

References

- https://geocompr.robinlovelace.net/geometric-operations.html
- https://bookdown.org/lexcomber/brunsdoncomber2e/Ch5.html