

Operations With Vector Data I

HES 505 Fall 2023: Session 11

Matt Williamson



Today's Plan

Objectives

By the end of today, you should be able to:

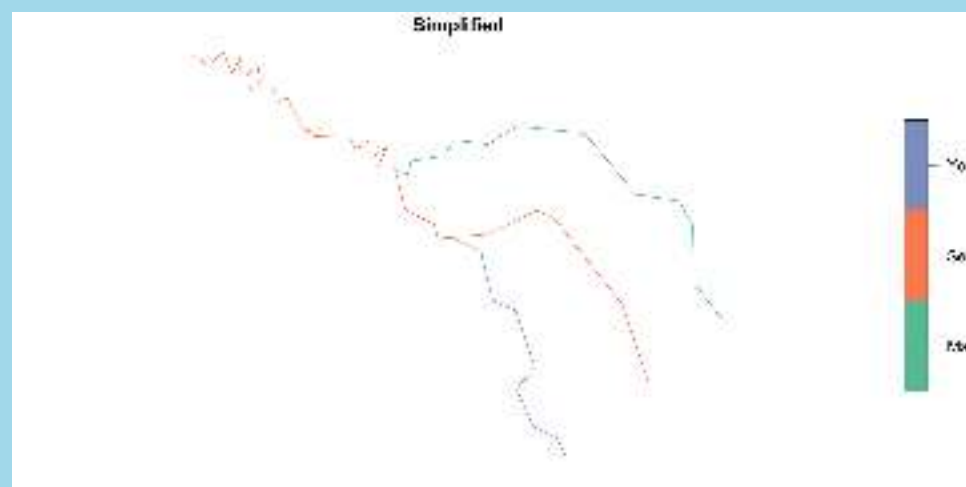
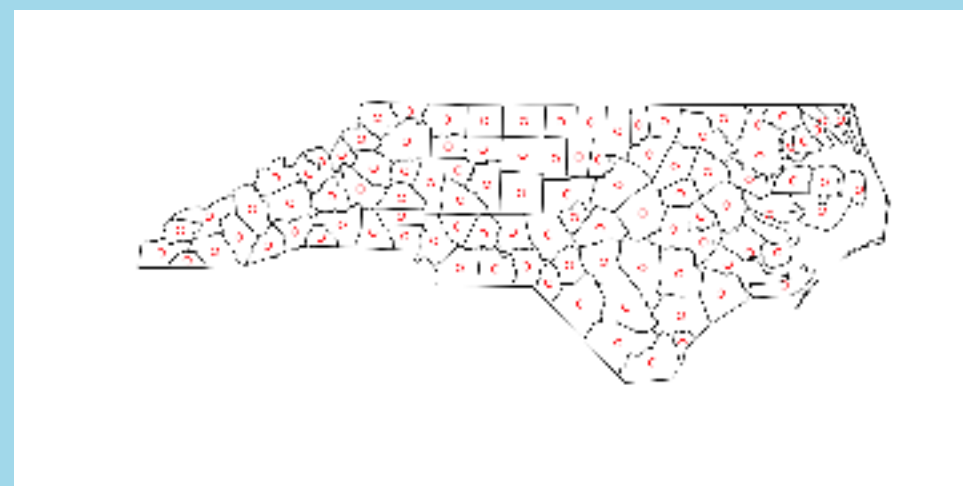
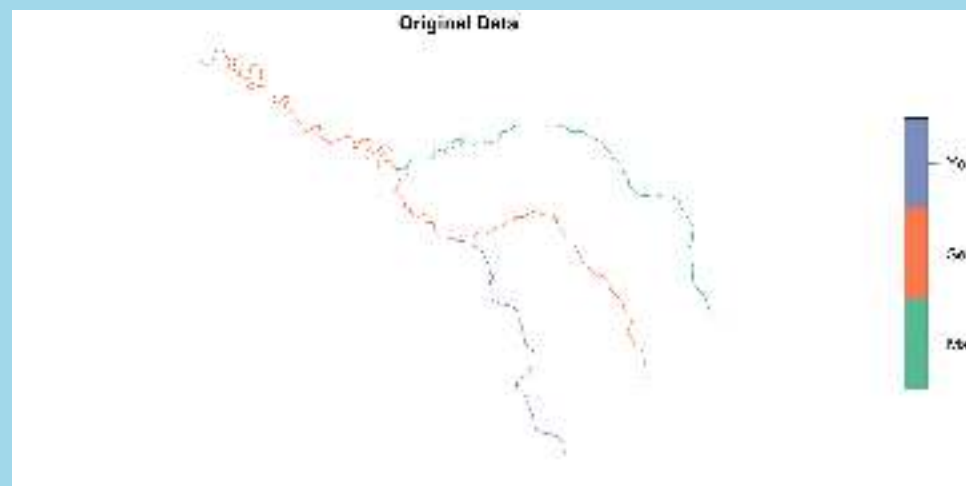
- Recognize the unary, binary, and n-ary transformers
- Articulate common uses for unary and binary transformers
- Use unary transformations to fix invalid geometries
- Implement common binary transformers to align and combine data

Revisiting **predicates** and **measures**

- **Predicates:** evaluate a logical statement asserting that a property is **TRUE**
- **Measures:** return a numeric value with units based on the units of the CRS
- Unary, binary, and n-ary distinguish how many geometries each function accepts and returns

Transformations

- **Transformations:** create new geometries based on input geometries



Unary Transformations

transformer	returns a geometry ...
<code>centroid</code>	of type POINT with the geometry's centroid
<code>buffer</code>	that is larger (or smaller) than the input geometry, depending on the buffer size
<code>jitter</code>	that was moved in space a certain amount, using a bivariate uniform distribution
<code>wrap_dateline</code>	cut into pieces that do no longer cover the dateline
<code>boundary</code>	with the boundary of the input geometry
<code>convex_hull</code>	that forms the convex hull of the input geometry
<code>line_merge</code>	after merging connecting LINESTRING elements of a MULTILINESTRING into longer LINESTRINGs .
<code>make_valid</code>	that is valid
<code>node</code>	with added nodes to linear geometries at intersections without a node; only works on individual linear geometries
<code>point_on_surface</code>	with a (arbitrary) point on a surface
<code>polygonize</code>	of type polygon, created from lines that form a closed ring

Common Unary Transformations

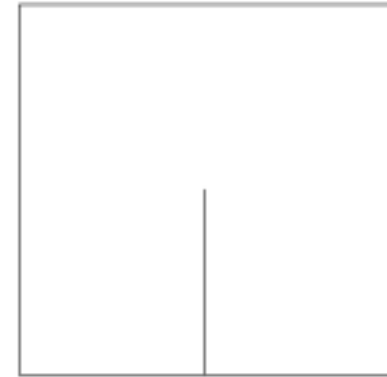
Fixing geometries

- When `all(st_is_valid(your.shapefile))` returns **FALSE**

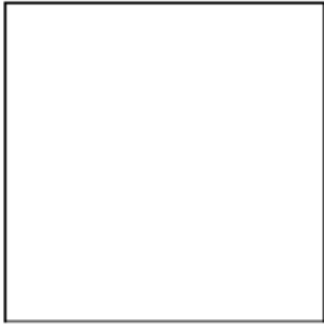
- `st_make_valid` has two methods:
 - original converts rings into noded lines and extracts polygons
 - structured makes rings valid first then merges/subtracts from existing polygons
 - Verify that the output is what you expect!!

```
1 ```{r}
2 x = st_sfc(st_polygon(list
3 st_is_valid(x)
4 ```
```

[1] FALSE



Fixing geometries with **st_make_valid**



```
1  ```{r}
2  y <- x %>% st_make_valid()
3  st_is_valid(y)
4  ```
```

```
[1] TRUE
```

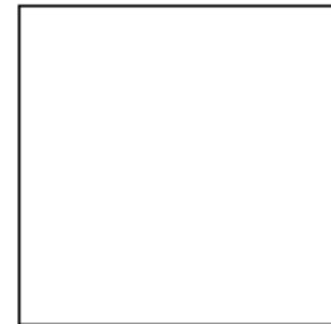
Fixing Geometries with `st_buffer`

-`st_buffer` enforces valid geometries as an output

- Setting a 0 distance buffer leaves most geometries unchanged
- Not all transformations do this

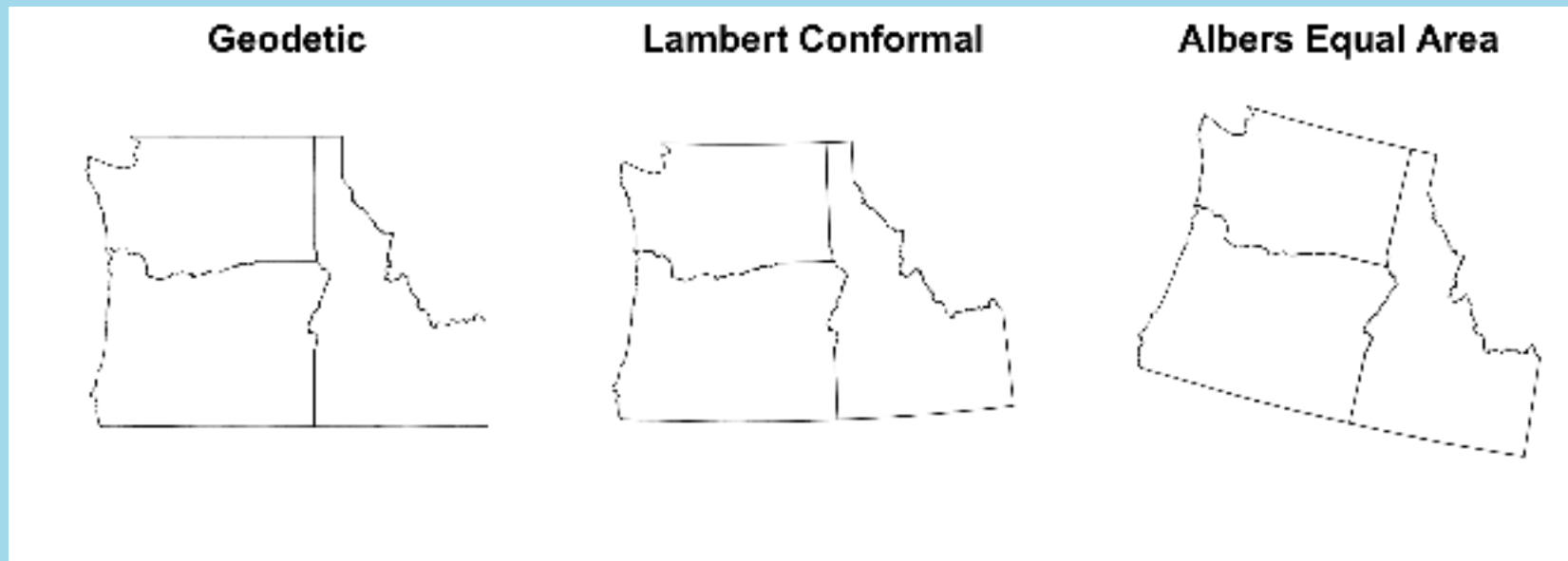
```
1  ```{r}
2  z <- x %>% st_buffer(., di
3
4  st_is_valid(z)
5  ```
```

```
[1] TRUE
```



Changing CRS with `st_transform`

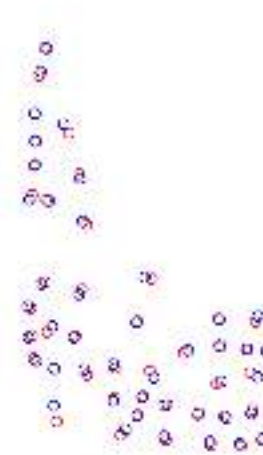
- You've already been using this!!
- Does not guarantee valid geometries (use `check = TRUE` if you want this)
- We'll try to keep things from getting too complicated



Converting areas to points with
st_centroid or
st_point_on_surface

- For “sampling” other datasets
- To simplify distance calculations
- To construct networks

```
1 id.counties <- tigris::counties(state = "ID", prog  
2 id.centroid <- st_centroid(id.counties)  
3 id.pointonsurf <- st_point_on_surface(id.counties)
```



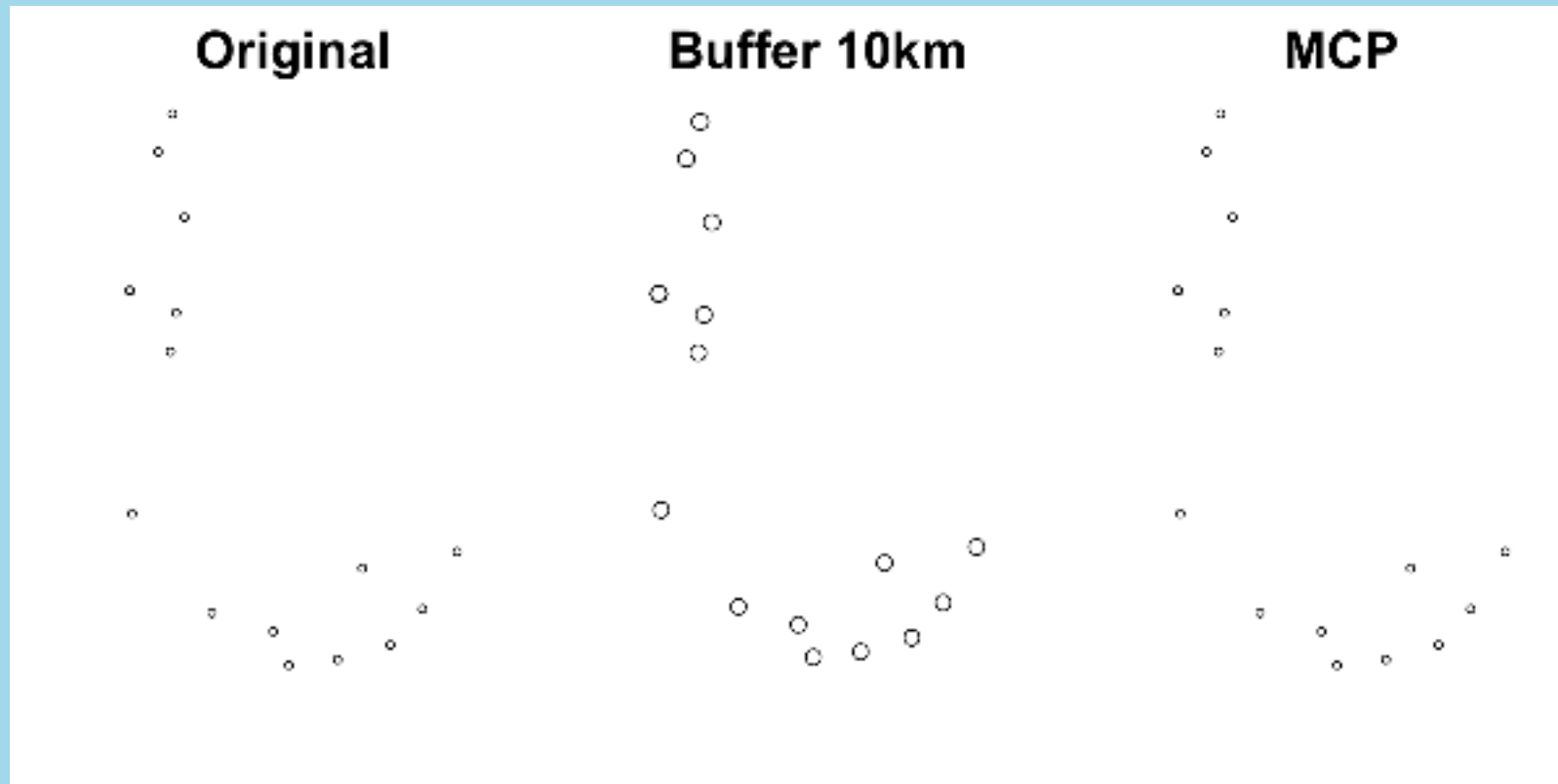
Creating “sampling areas”

- Uncertainty in your point locations
- Incorporate a fixed range around each point
- Combine multiple points into a single polygon

```
1 hospitals.id <- landmarks.id.csv %>%  
2   st_as_sf(., coords = c("longitude", "latitude")) %>%  
3   filter(., MTFCC == "K1231")  
4 st_crs(hospitals.id) <- 4326
```


Creating sampling areas

```
1 hospital.buf <- hospitals.id %>%  
2   st_buffer(., dist=10000)  
3  
4 hospital.mcp <- hospitals.id %>%  
5   st_convex_hull(.)
```



Other Unary Transformations

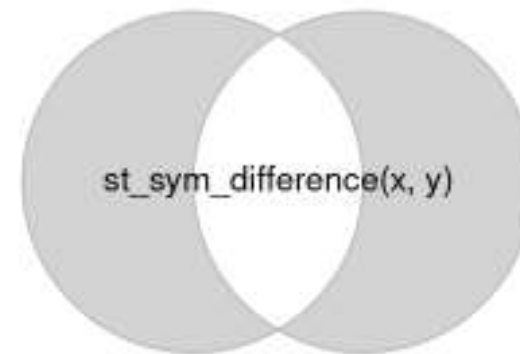
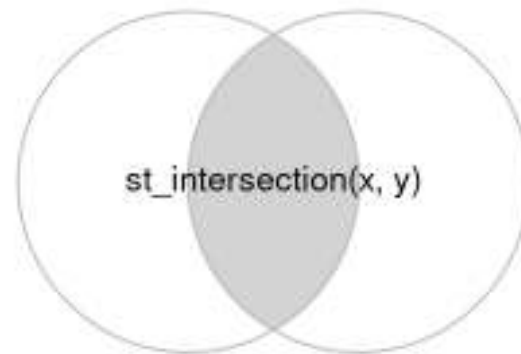
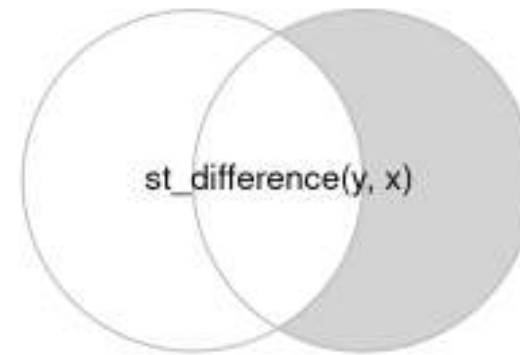
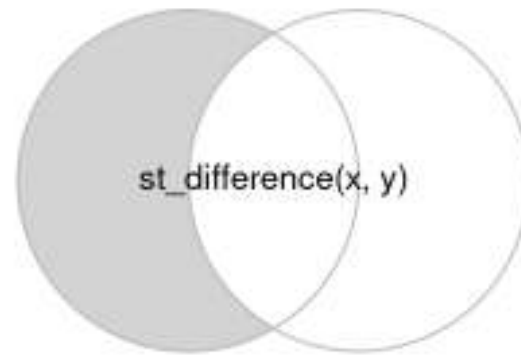
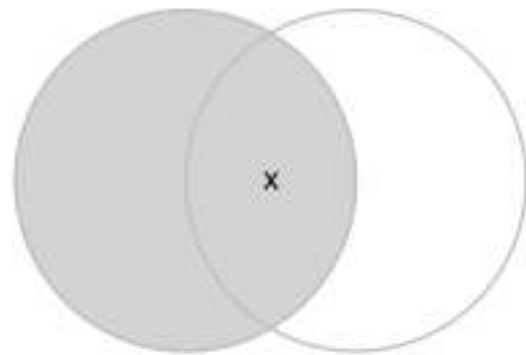
transformer	returns a geometry ...
<code>segmentize</code>	a (linear) geometry with nodes at a given density or minimal distance
<code>simplify</code>	simplified by removing vertices / nodes (lines or polygons)
<code>split</code>	that has been split with a splitting linestring
<code>transform</code>	transformed or convert to a new coordinate reference system (chapter @ref(cs))
<code>triangulate</code>	with Delauney triangulated polygon(s) (figure @ref(fig:vor))
<code>voronoi</code>	with the Voronoi tessellation of an input geometry (figure @ref(fig:vor))
<code>zm</code>	with removed or added Z and / or M coordinates
<code>collection_extract</code>	with subgeometries from a GEOMETRYCOLLECTION of a particular type
<code>cast</code>	that is converted to another type
<code>+</code>	that is shifted over a given vector
<code>*</code>	that is multiplied by a scalar or matrix

Binary Transformers

Binary Transformers

function	returns	infix operator
<code>intersection</code>	the overlapping geometries for pair of geometries	<code>&</code>
<code>union</code>	the combination of the geometries; removes internal boundaries and duplicate points, nodes or line pieces	<code> </code>
<code>difference</code>	the geometries of the first after removing the overlap with the second geometry	<code>/</code>
<code>sym_difference</code>	the combinations of the geometries after removing where they intersect; the negation (opposite) of <code>intersection</code>	<code>%/%</code>
<code>crop</code>	crop an sf object to a specific rectangle	

Binary Transformers



Common Uses of Binary Transformers

- Relating partially overlapping datasets to each other
- Reducing the extent of vector objects

N-ary Transformers

- Similar to Binary (except `st_crop`)
- `union` can be applied to a set of geometries to return its geometrical union
- `intersection` and `difference` take a single argument, but operate (sequentially) on all pairs, triples, quadruples, etc.

