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:

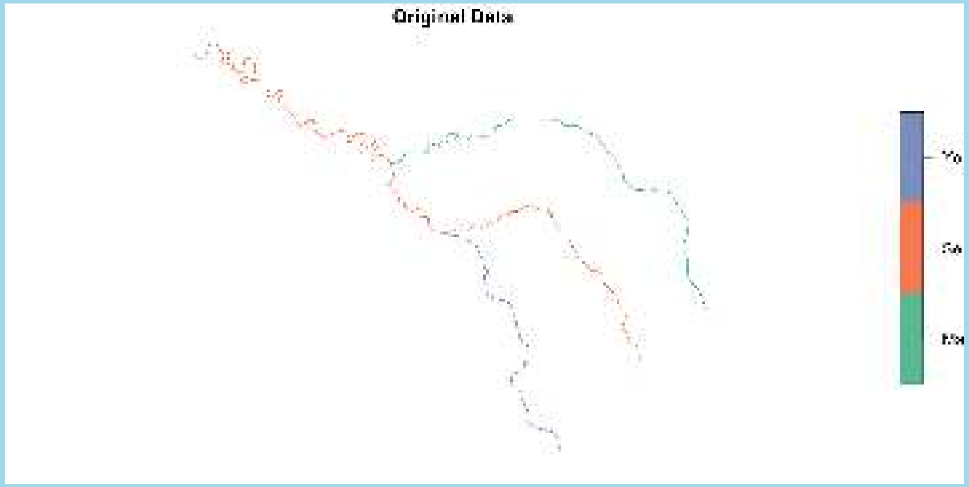
- Recognize the unary, binary, and n-ary transformers
- Articulate common uses for unary and binary transformers
- Use unary transformations to fix invalid geometries
- Implement common binary transformers to align and combine data

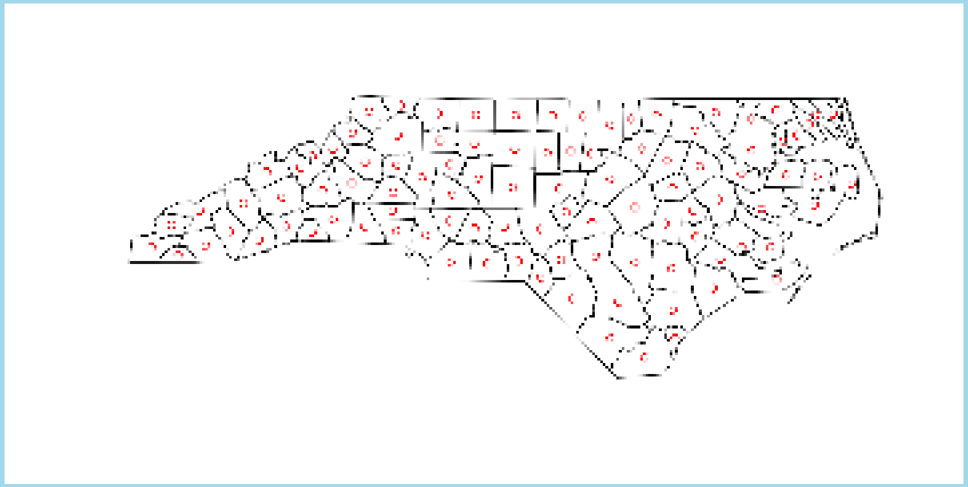
Revisiting `predicates` and `measures`

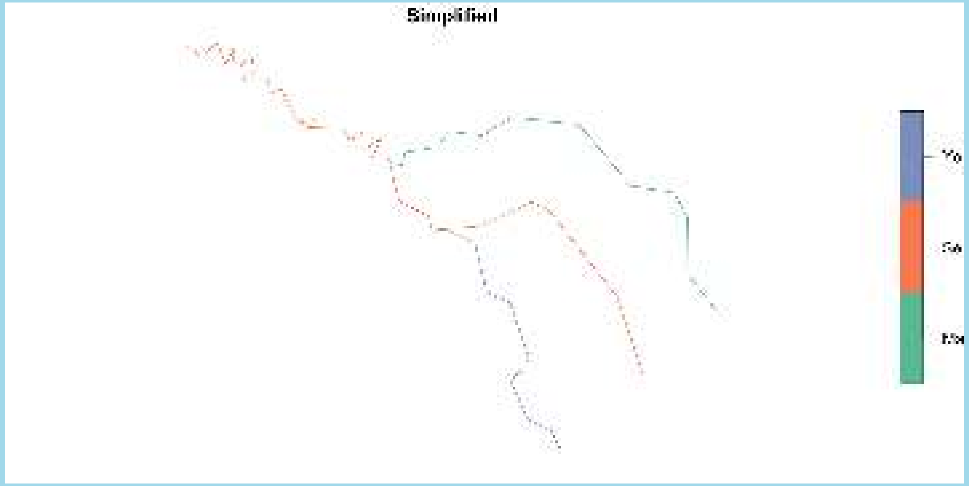
- **Predicates**: evaluate a logical statement asserting that a property is `TRUE`
- **Measures**: return a numeric value with units based on the units of the CRS
- Unary, binary, and n-ary distinguish how many geometries each function accepts and returns

Transformations

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


Original Data
Yo
Sa
Ma




Simplified
Yo
Sa
Ma

Unary Transformations

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

Common Unary Transformations

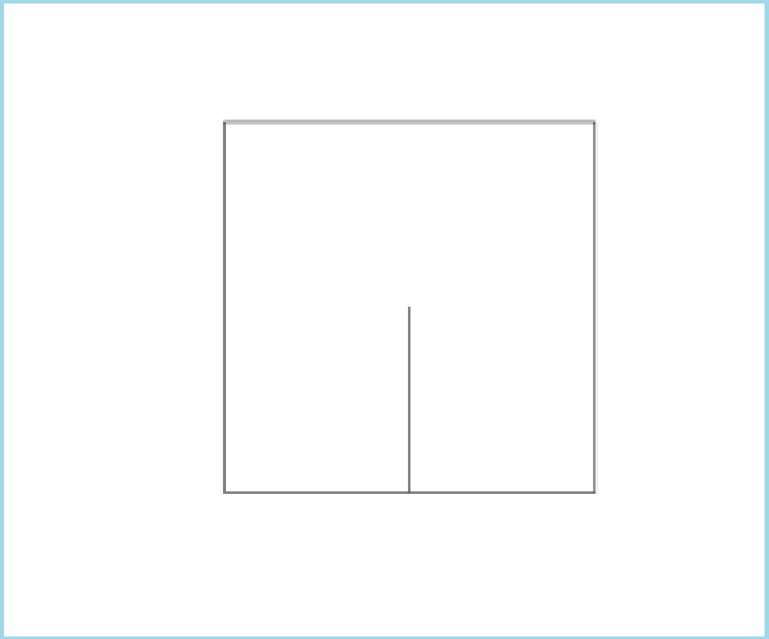
Fixing geometries

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

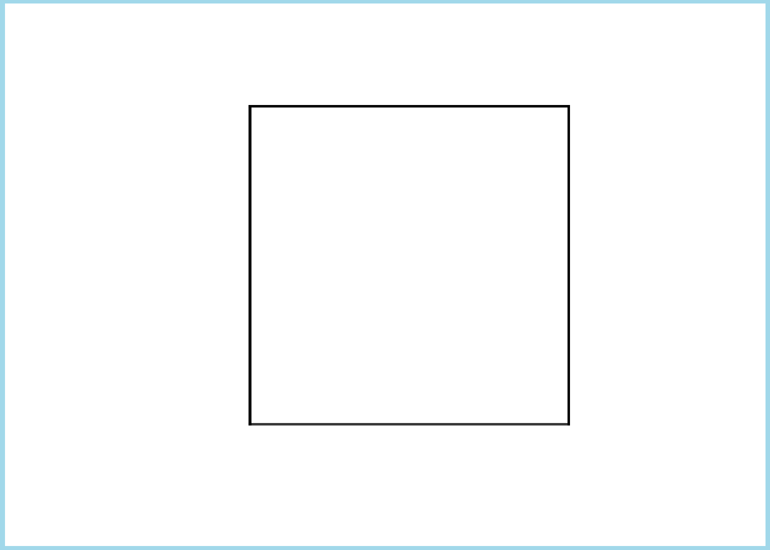
- `st_make_valid` has two methods:
  - original converts rings into noded lines and extracts polygons
  - structured makes rings valid first then merges/subtracts from existing polgyons
  - Verify that the output is what you expect!!

~~~
```{r}
x = st_sfc(st_polygon(list
st_is_valid(x)
```
~~~

```
[1] FALSE
```



Fixing geometries with `st_make_valid`



~~~
```{r}
y <- x %>% st_make_valid()
st_is_valid(y)
```
~~~

```
[1] TRUE
```

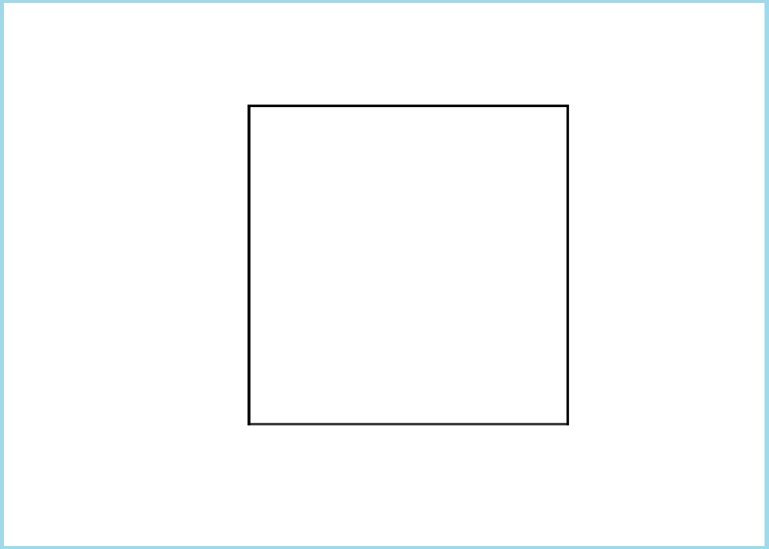
Fixing Geometries with `st_buffer`

-`st_buffer` enforces valid geometries as an output

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

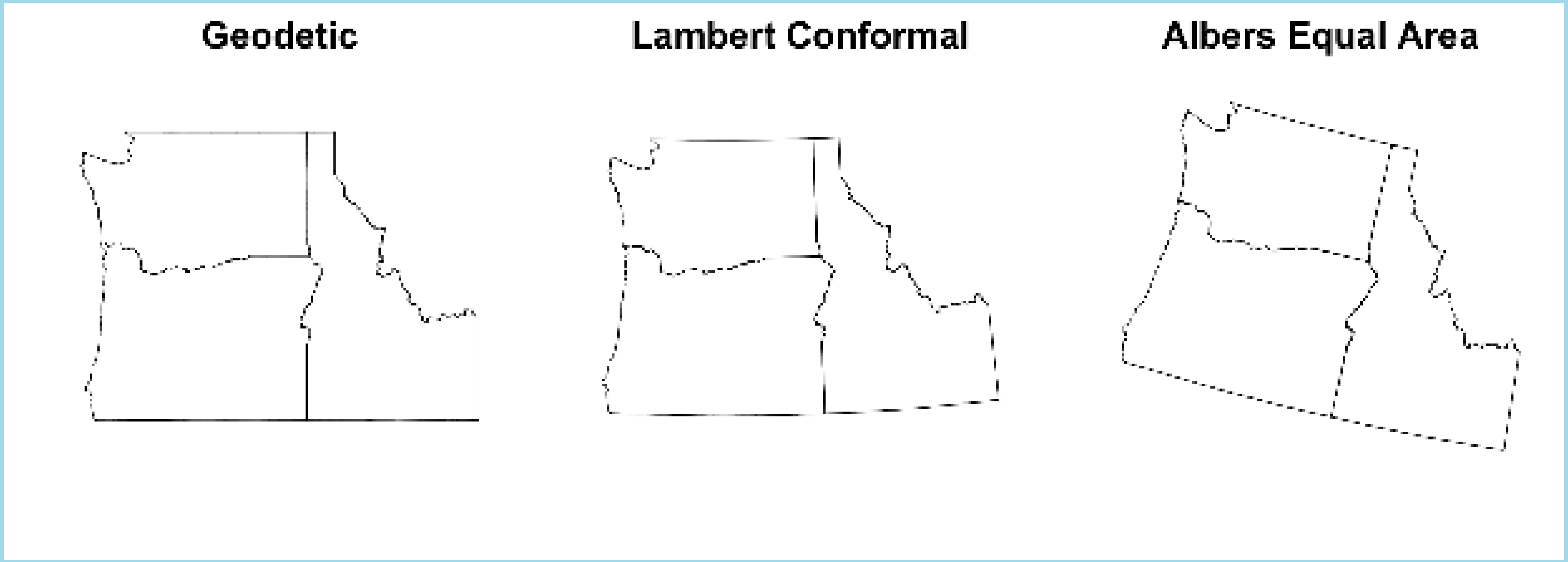
~~~
```{r}
z <- x %>% st_buffer(., di

st_is_valid(z)
```
~~~

```
[1] TRUE
```



Changing `CRS` with `st_transform`

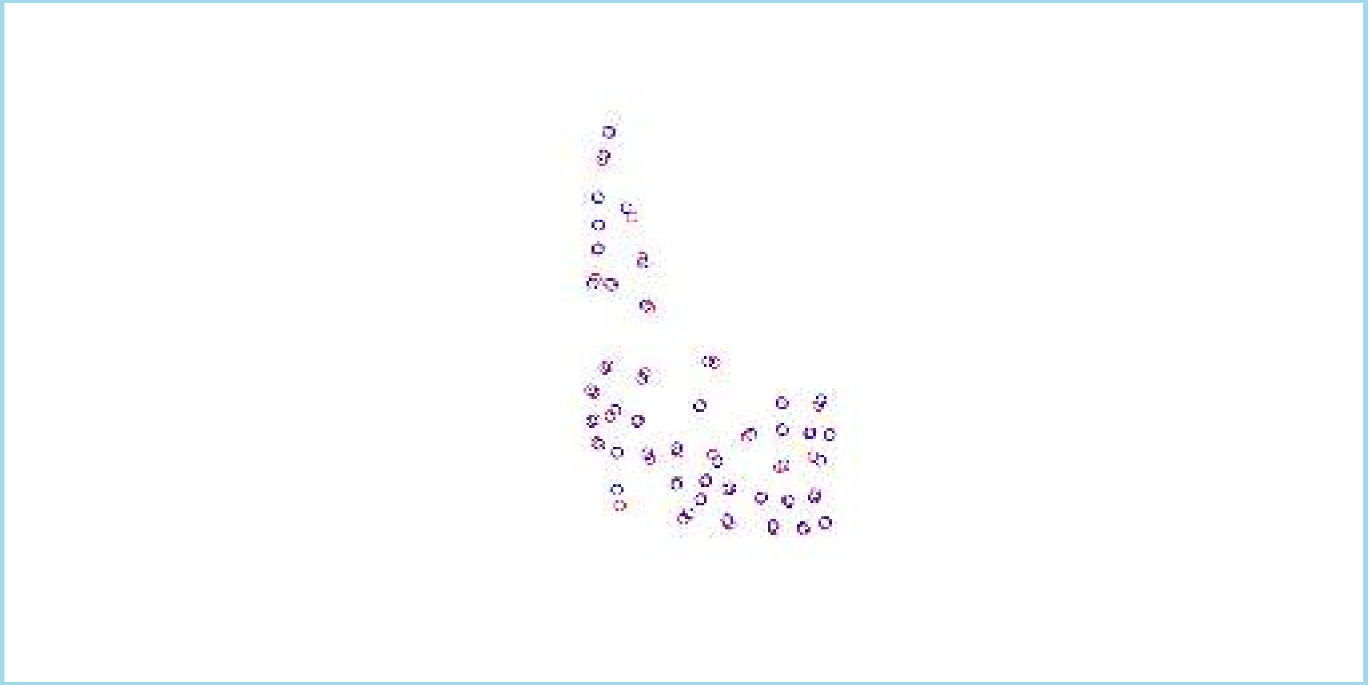
- You've already been using this!!
- Does not guarantee valid geometries (use `check = TRUE` if you want this)
- We'll try to keep things from getting too complicated



Converting areas to points with `st_centroid` or `st_point_on_surface`

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

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



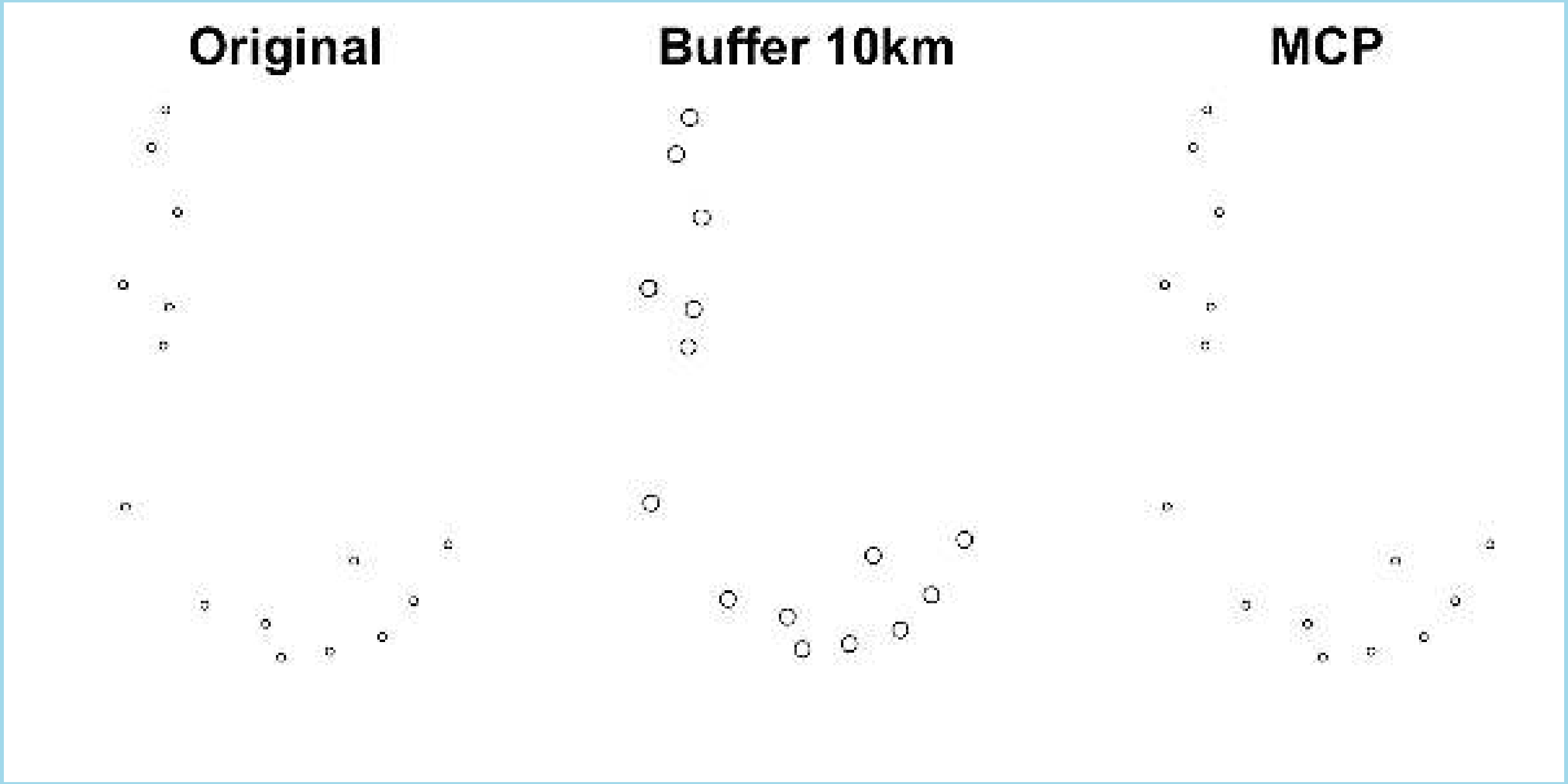
Creating "sampling areas"

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

```r
hospitals.id <- landmarks.id.csv %>%
  st_as_sf(., coords = c("longitude", "lattitude")) %>%
  filter(., MTFCC == "K1231")
st_crs(hospitals.id) <- 4326
```

Creating sampling areas

```
hospital.buf <- hospitals.id %>%
  st_buffer(., dist=10000)

hospital.mcp <- hospitals.id %>%
  st_convex_hull(.)
```



Other Unary Transformations

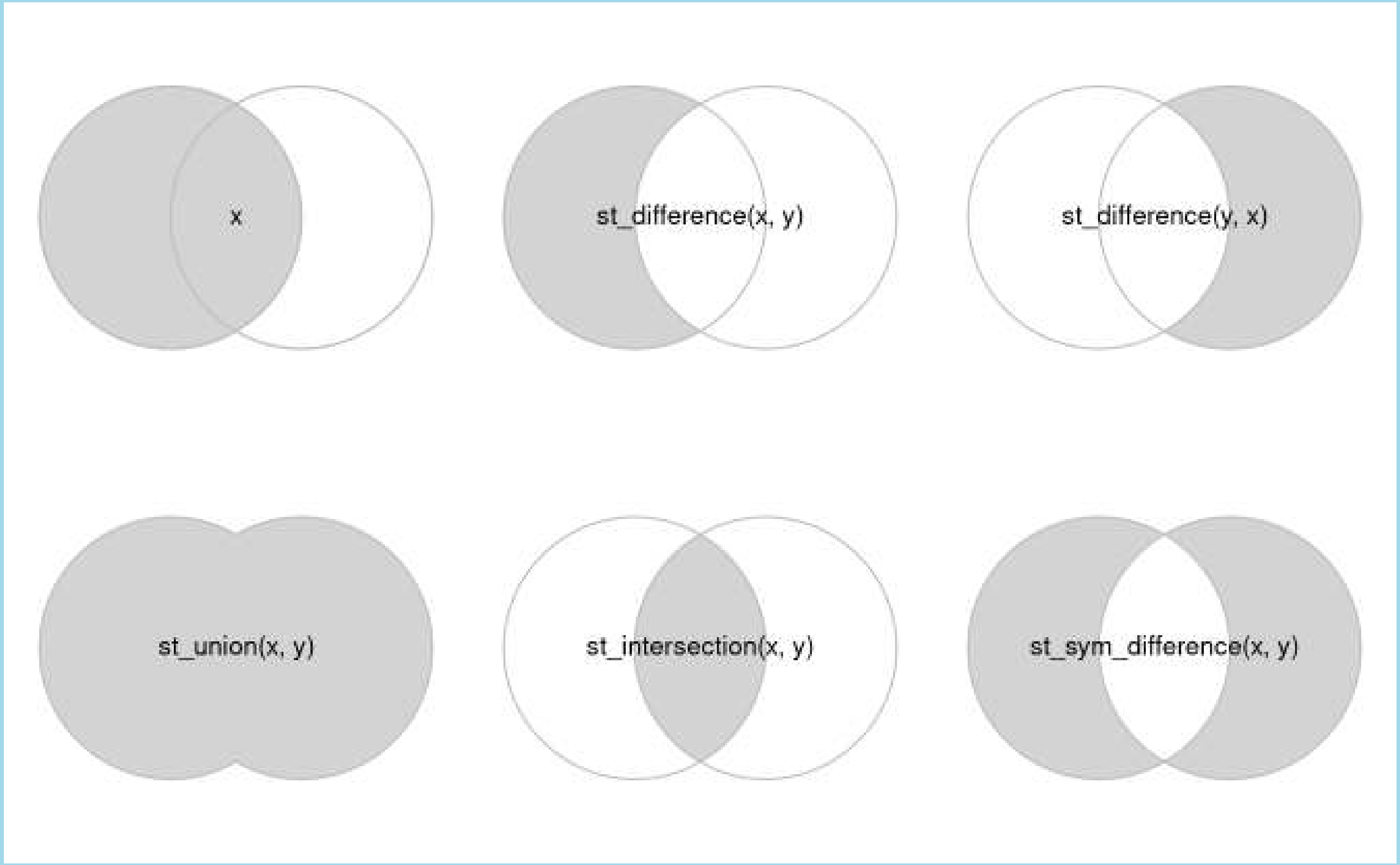
transformer	**returns a geometry …**
`segmentize`	a (linear) geometry with nodes at a given density or minimal distance
`simplify`	simplified by removing vertices/nodes (lines or polygons)
`split`	that has been split with a splitting linestring
`transform`	transformed or convert to a new coordinate reference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

Binary Transformers

Binary Transformers

function	returns	infix operator
`intersection`	the overlapping geometries for pair of geometries	`&`
`union`	the combination of the geometries; removes internal boundaries and duplicate points, nodes or line pieces	`|`
`difference`	the geometries of the first after removing the overlap with the second geometry	`/`
`sym_difference`	the combinations of the geometries after removing where they intersect; the negation (opposite) of `intersection`	`%/%`
`crop`	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.