Operations With Vector Data I

HES 505 Fall 2024: Session 10

Carolyn Koehn

Announcements

Due this week: assignment revision 1

Today's Plan

Objectives

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

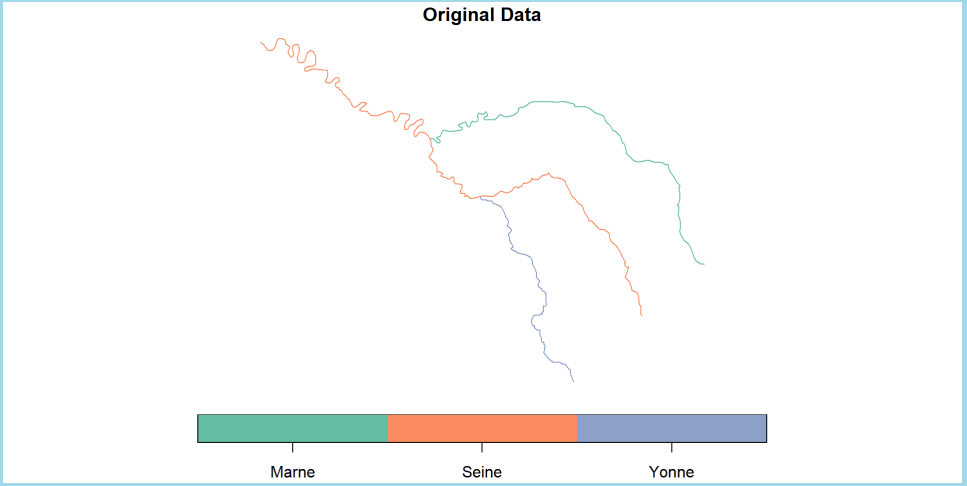
- Recognize the unary, binary, and n-ary transformers
- Articulate common uses for unary and binary transformers
- Use unary transformations to fix invalid geometries
- Implement common binary transformers to align and combine data

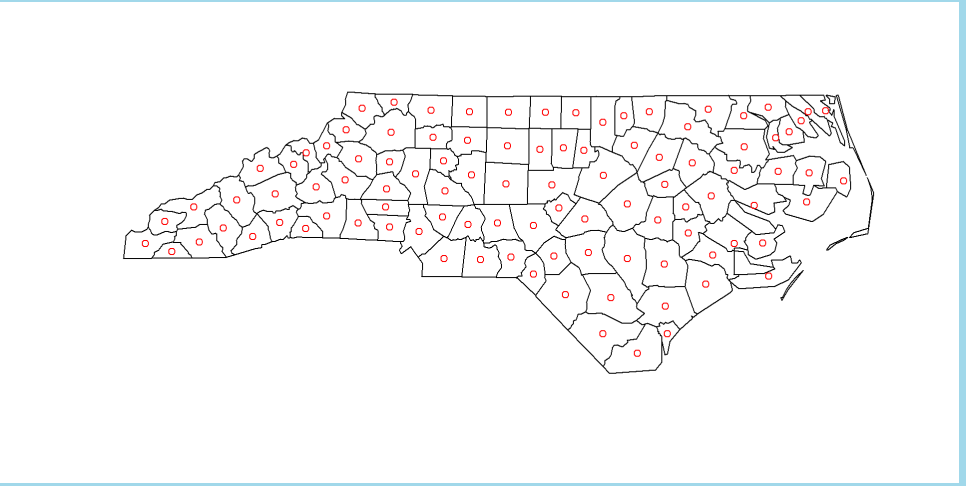
Revisiting `predicates` and `measures`

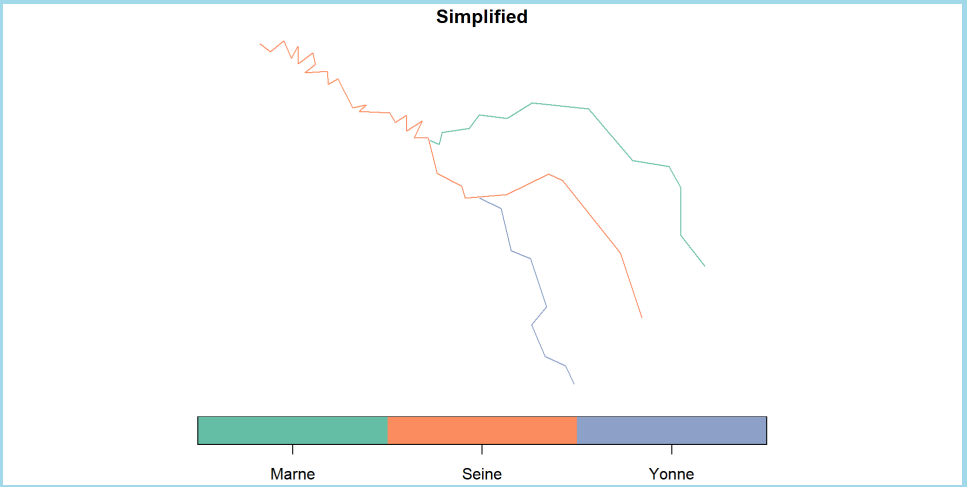
- **Predicates**: evaluate a logical statement asserting that a property is `TRUE`
- **Measures**: return a numeric value with units based on the units of the CRS
- Unary, binary, and n-ary distinguish how many geometries each function accepts and returns

Transformations

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







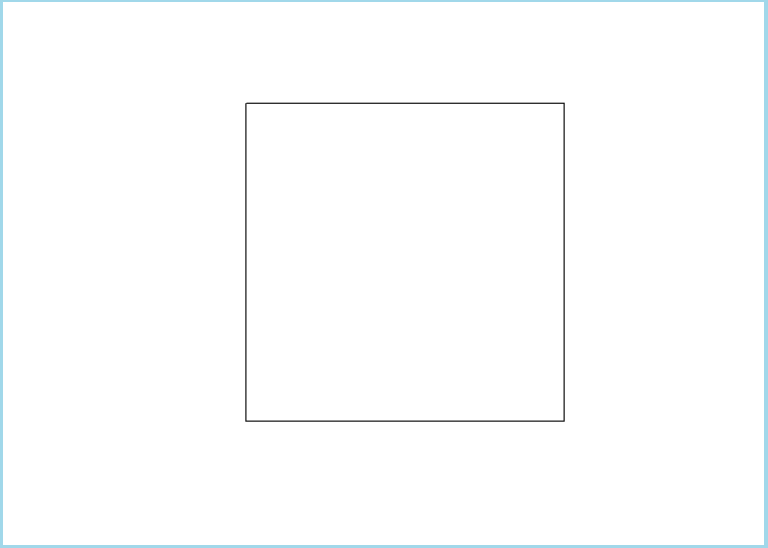
Unary Transformations

transformer	returns a geometry …
`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 an arbitrary point on a surface
`polygonize`	of type polygon, created from lines that form a closed ring

Common Unary Transformations

Fixing geometries

Fixing geometries with `st_make_valid`



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

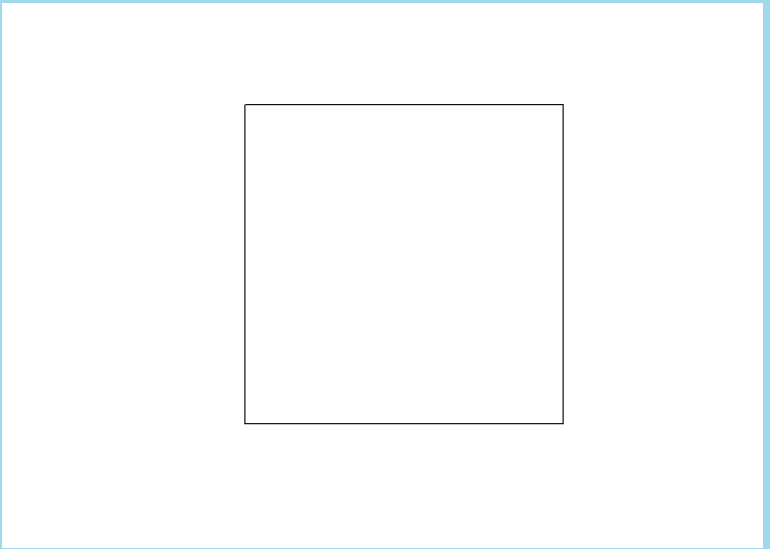
```
[1] TRUE
```

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

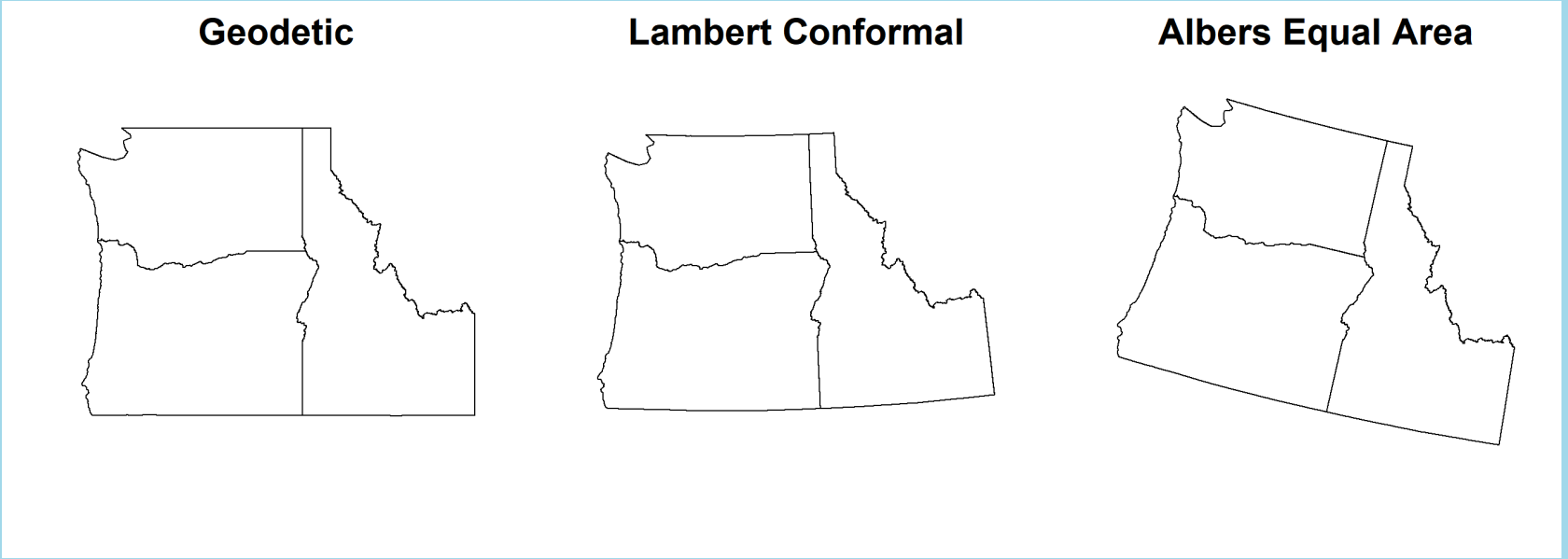
```
1  ```{r}
2  z <- x %>% st_buf
3
4  st_is_valid(z)
5  ```
```

```
[1] TRUE
```



Changing CRS with st_transform

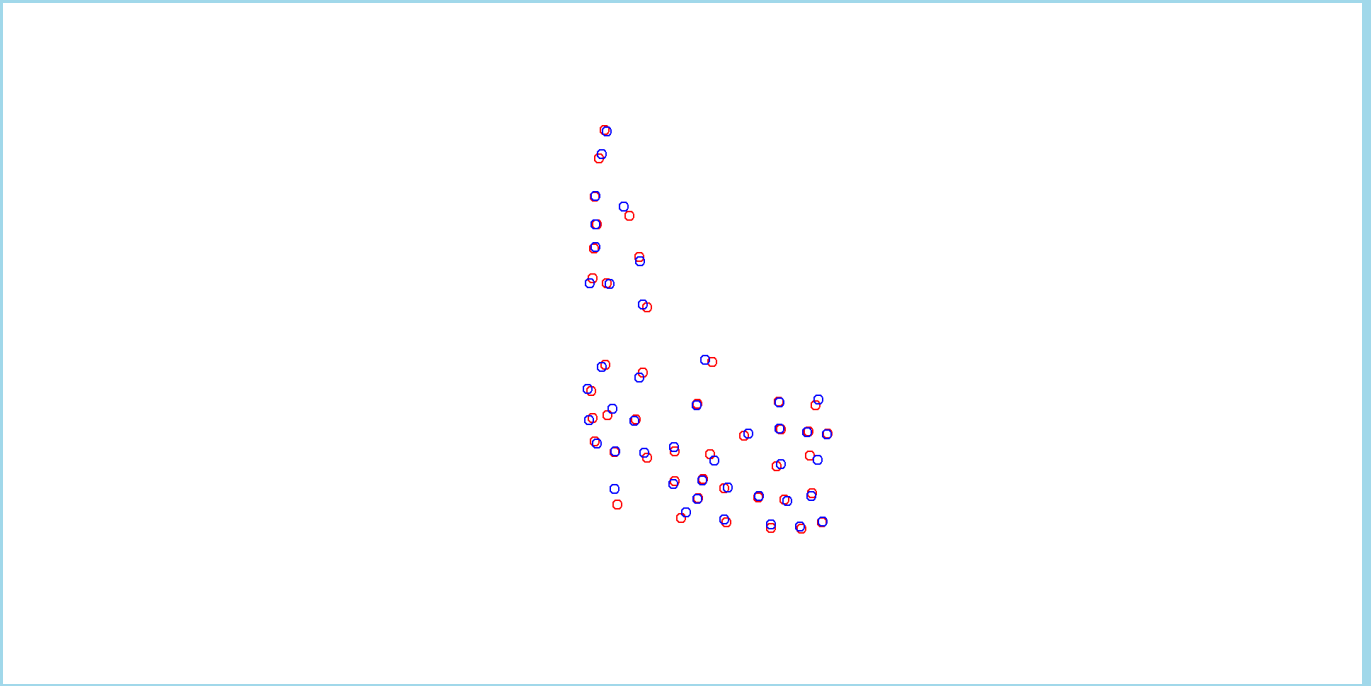
- You've already been using this!!
- Does not guarantee valid geometries (use `check = TRUE` if you want this)
- We'll try to keep things from getting too complicated



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 = "
2 id.centroid <- st_centroid(id.counties)
3 id.pointonsurf <- st_point_on_surface(id.
```



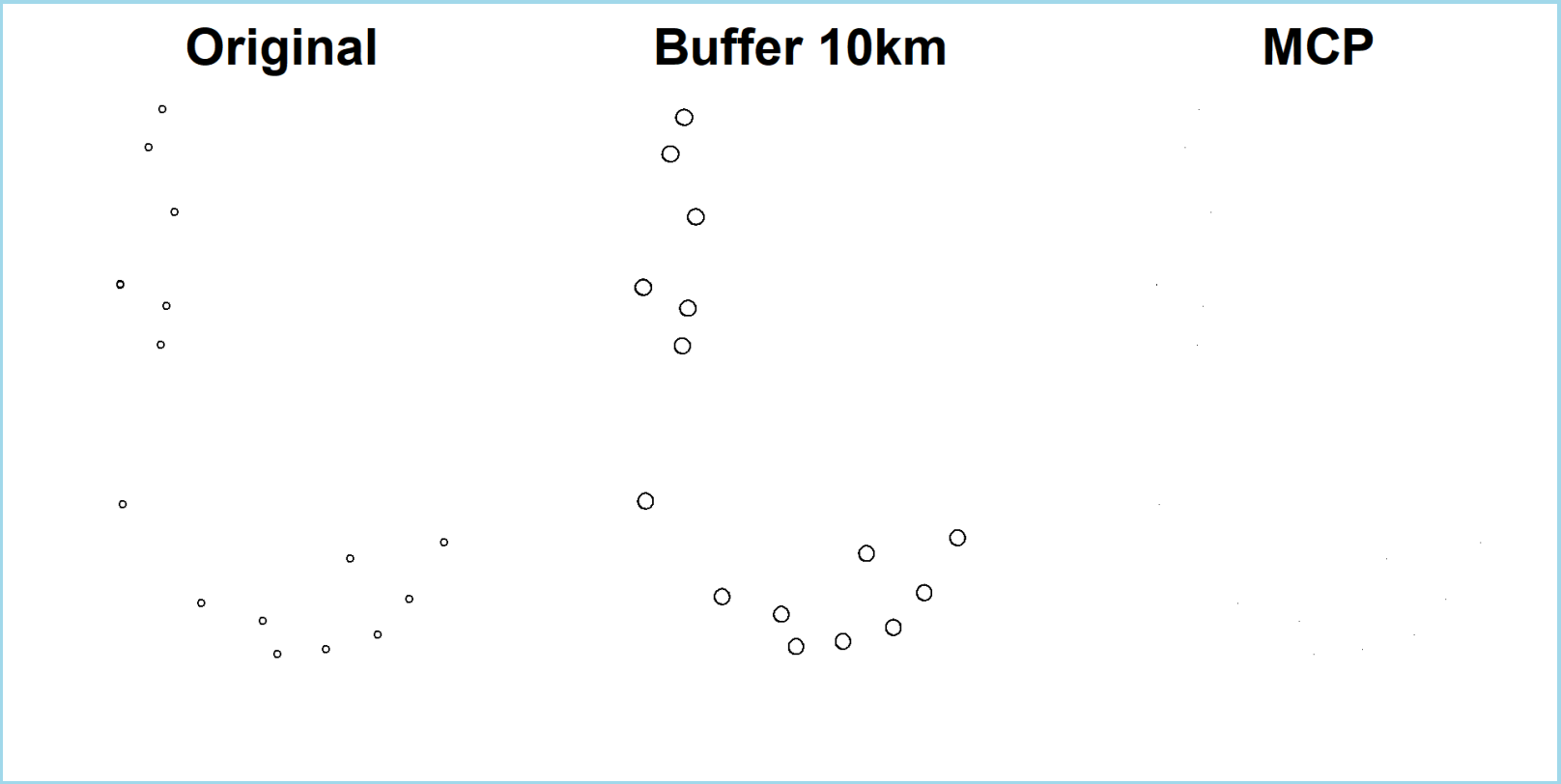
Creating “sampling areas”

- 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", "lattitude")) %>%
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

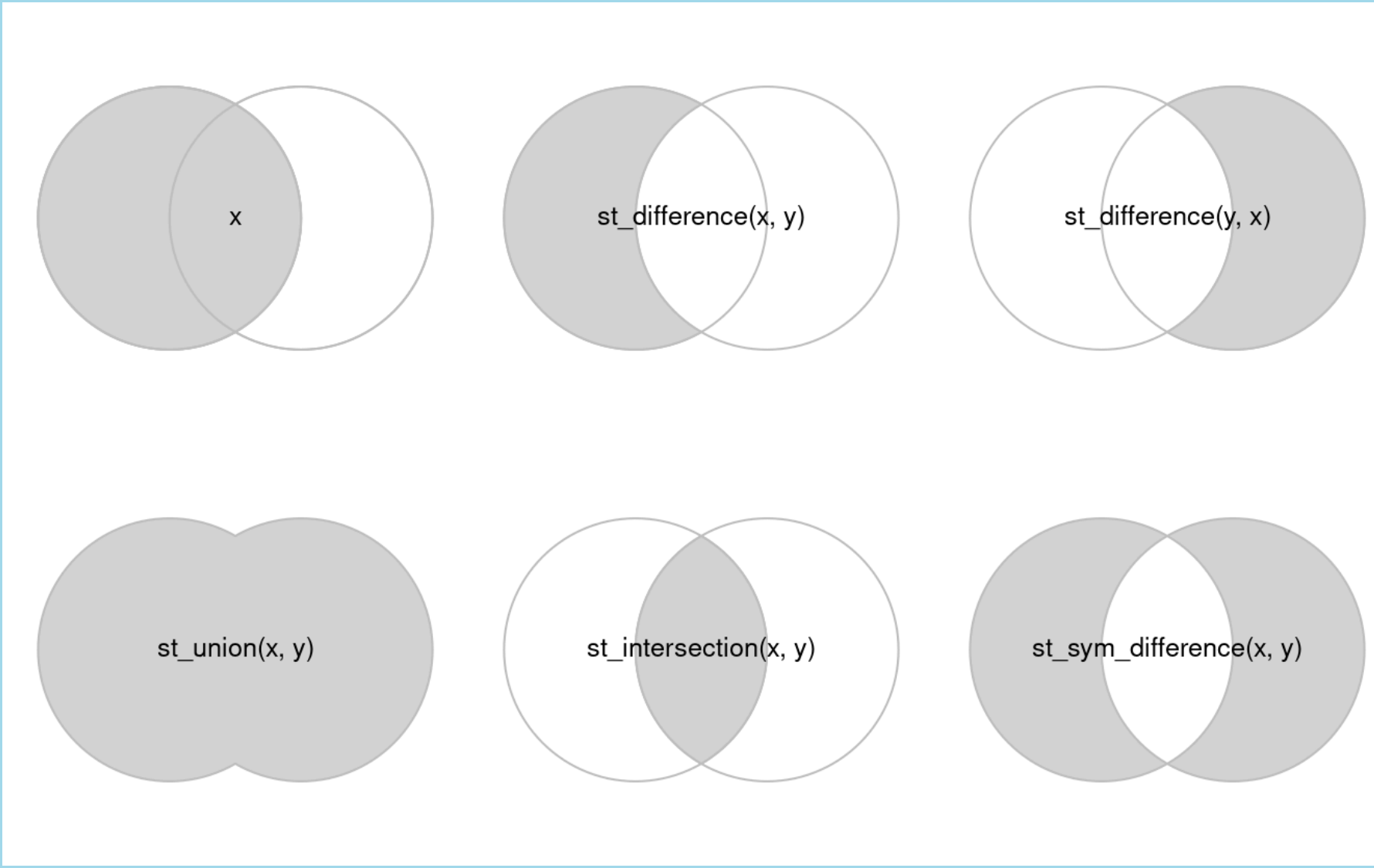
transformer	returns a geometry …
`segmentize`	a (linear) geometry with nodes at a given density or minimal distance
`simplify`	simplified by removing vertices/nodes (lines or polygons)
`split`	that has been split with a splitting linestring
`transform`	transformed or convert to a new coordinate reference system
`triangulate`	with Delauney triangulated polygon(s)
`voronoi`	with the Voronoi tessellation of an input geometry
`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.

Practice

Centroids and Distances

The function `system.time` tells you how long a function takes to run:

```
1 system.time(id.counties <- tigris::counties(state = "ID", progress_
```

```
   user  system elapsed
   1.02    0.30    1.33
```

Find the counties that are the furthest distance from each other in Idaho using the polygons, centroids, and point on surface objects we created earlier. Which distance calculation is the fastest?

(You may want to refer to our session 7 example code.)

Intersections and Buffers

`tigris::primary_secondary_roads()` retrieves shapefiles for major roads in each state of the US.

1. Plot just Ada county and the major roads within.
2. Map the portion of major roads that are within 50 km (as the crow flies) of the center of Ada county. (Remember to check the units of your CRS.)
3. Challenge: Include county borders in your plot for part 2.