Spatial Data as Vectors

HES 505 Fall 2022: Session 7

Matt Williamson


Map
OF THE MINING SECTIONS OF
IDAHO & OREGON
embracing the GOLD and SILVER mines of
BOISE & OWYHEE
BY GEO. WOODMAN
compiled chiefly from notes of his travels and Surveys during the last 18 months.
PUBLISHED BY A. GENSOUL
58 Montgomery St. San Francisco.
REFERENCE.
Wagon Roads.
Trails.
Railroads.
Towns and Ranches
Forts.
Springs.
Sage Desert
Humboldt Mines
Silver
South line of Idaho Terr.
Salt Lake City
OREGON
IDAHO
NEVADA
CALIFORNIA

Today's Plan

Objectives

- Articulate the role of the data model in geographic information systems
- Describe the key elements of vector data
- Use the `sf` package to read and manipulate vector data
- Define `geometry` in the context of vector objects and troubleshoot common problems

What is a data model?

- **Data:** a collection of discrete values that describe phenomena
- Your brain stores millions of pieces of data
- Computers are not your brain
  - Need to organize data systematically
  - Be able to display and access efficiently
  - Need to be able to store and access repeatedly
- Data models solve this problem

2 Types of Spatial Data Models

- **Raster:** grid-cell tessellation of an area. Each raster describes the value of a single phenomenon. More next week…
- **Vector:** (many) attributes associated with locations defined by coordinates

The Vector Data Model

- **Vertices** (i.e., discrete x-y locations) define the shape of the vector
- The organization of those vertices define the *shape* of the vector
- General types: points, lines, polygons

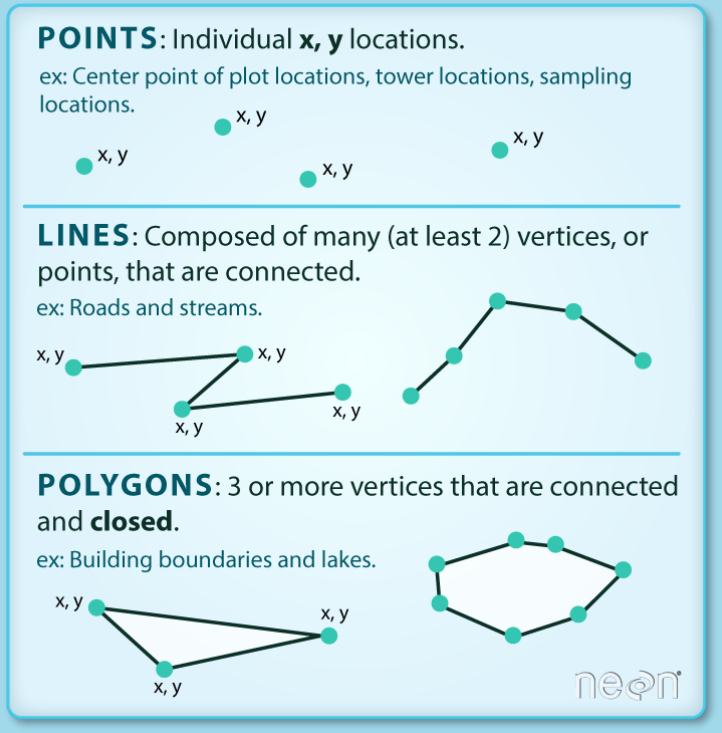
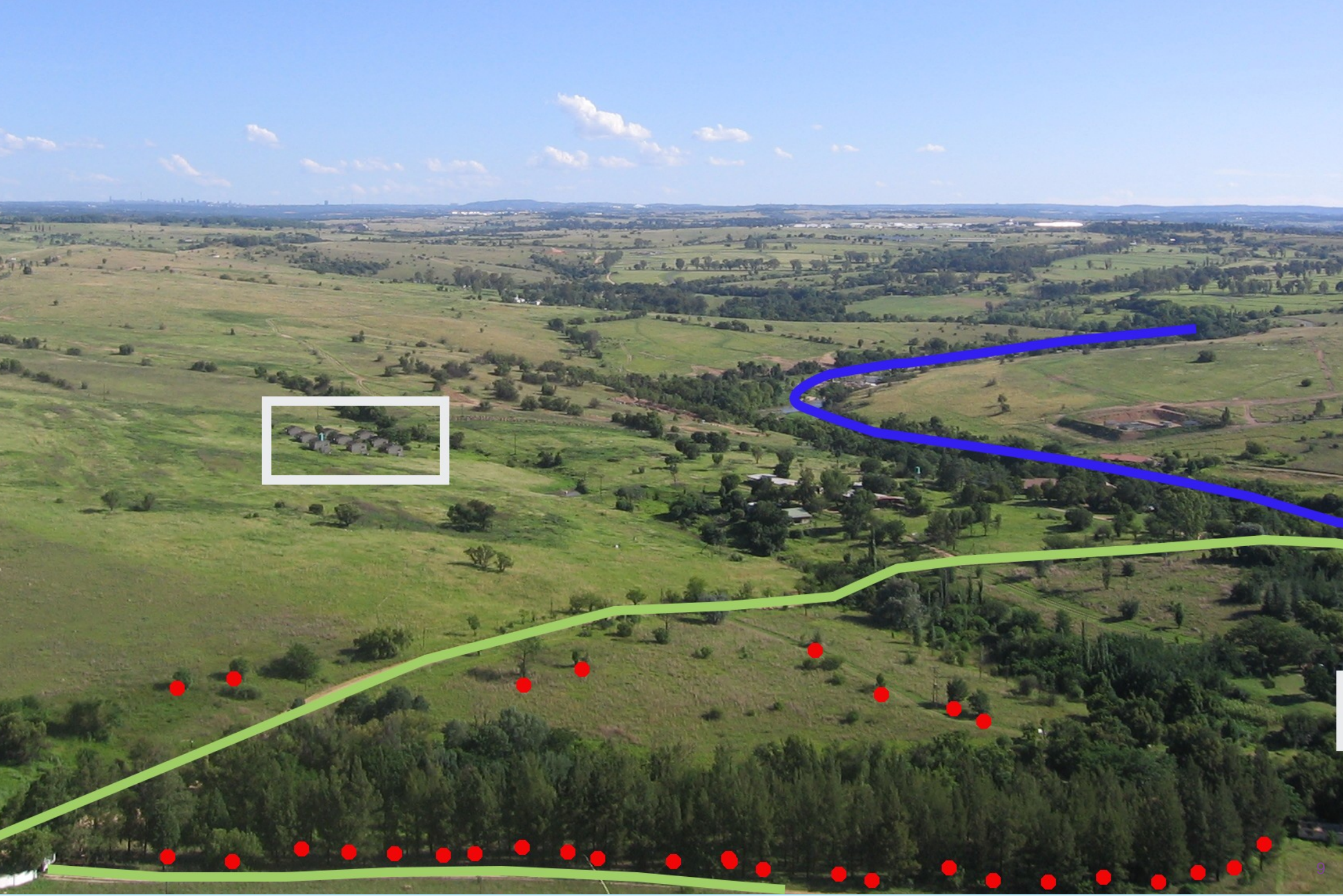
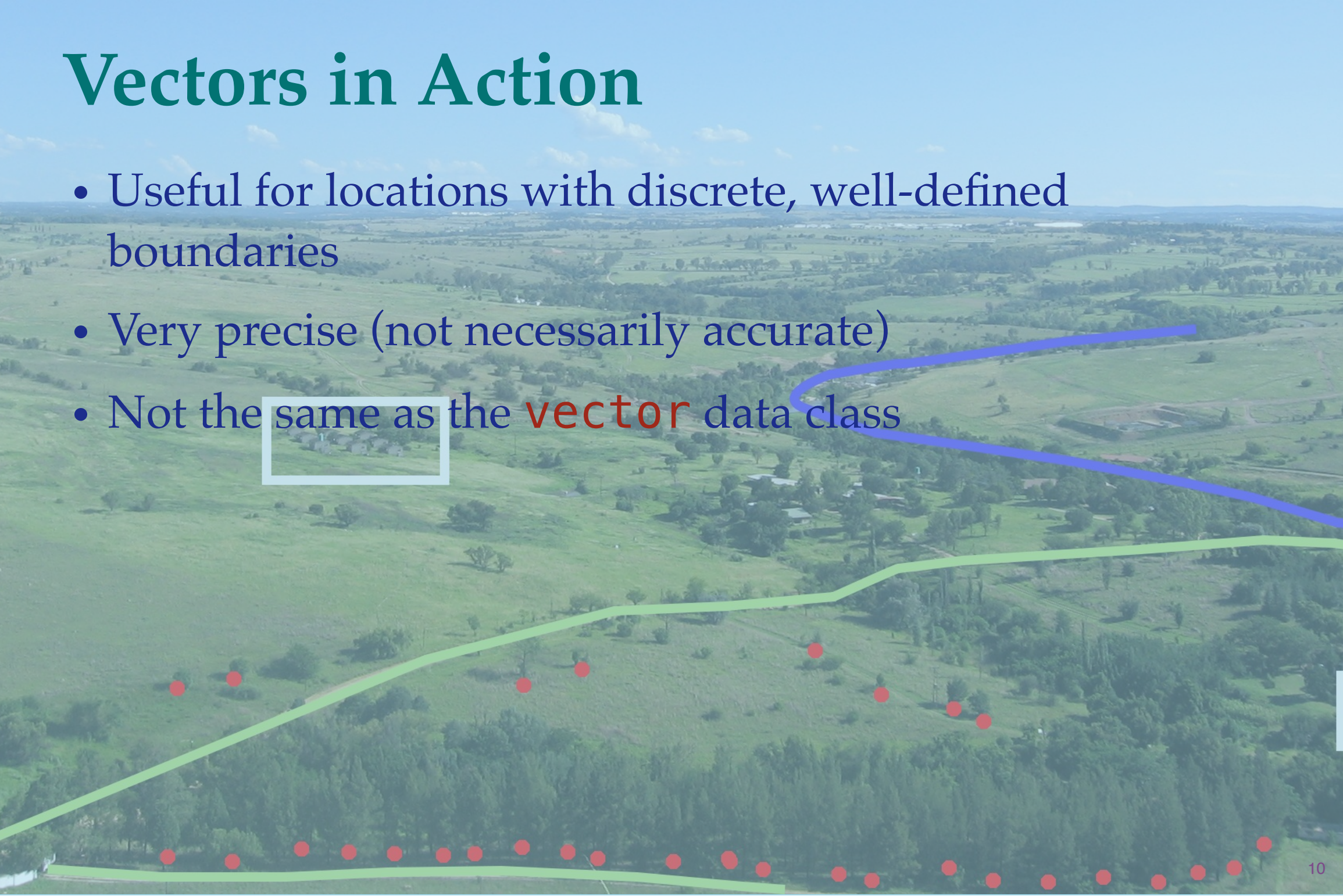


Image Source: Colin Williams (NEON)



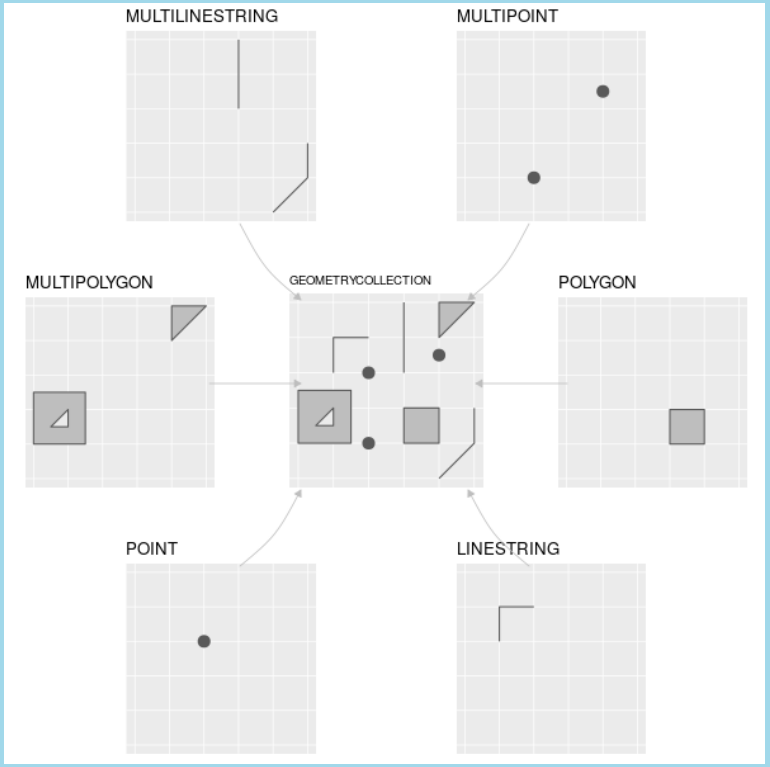
Vectors in Action

- Useful for locations with discrete, well-defined boundaries
- Very precise (not necessarily accurate)
- Not the same as the `vector` data class



Vectors in R

Representing vector data in R



From Lovelace et al.

- `sf` hierarchy reflects increasing complexity of geometry
  - `st_point`, `st_linestring`, `st_polygon` for single features
  - `st_multi*` for multiple features of the same type
  - `st_geometrycollection` for multiple feature types
  - `st_as_sfc` creates the geometry list column for many `sf` operations

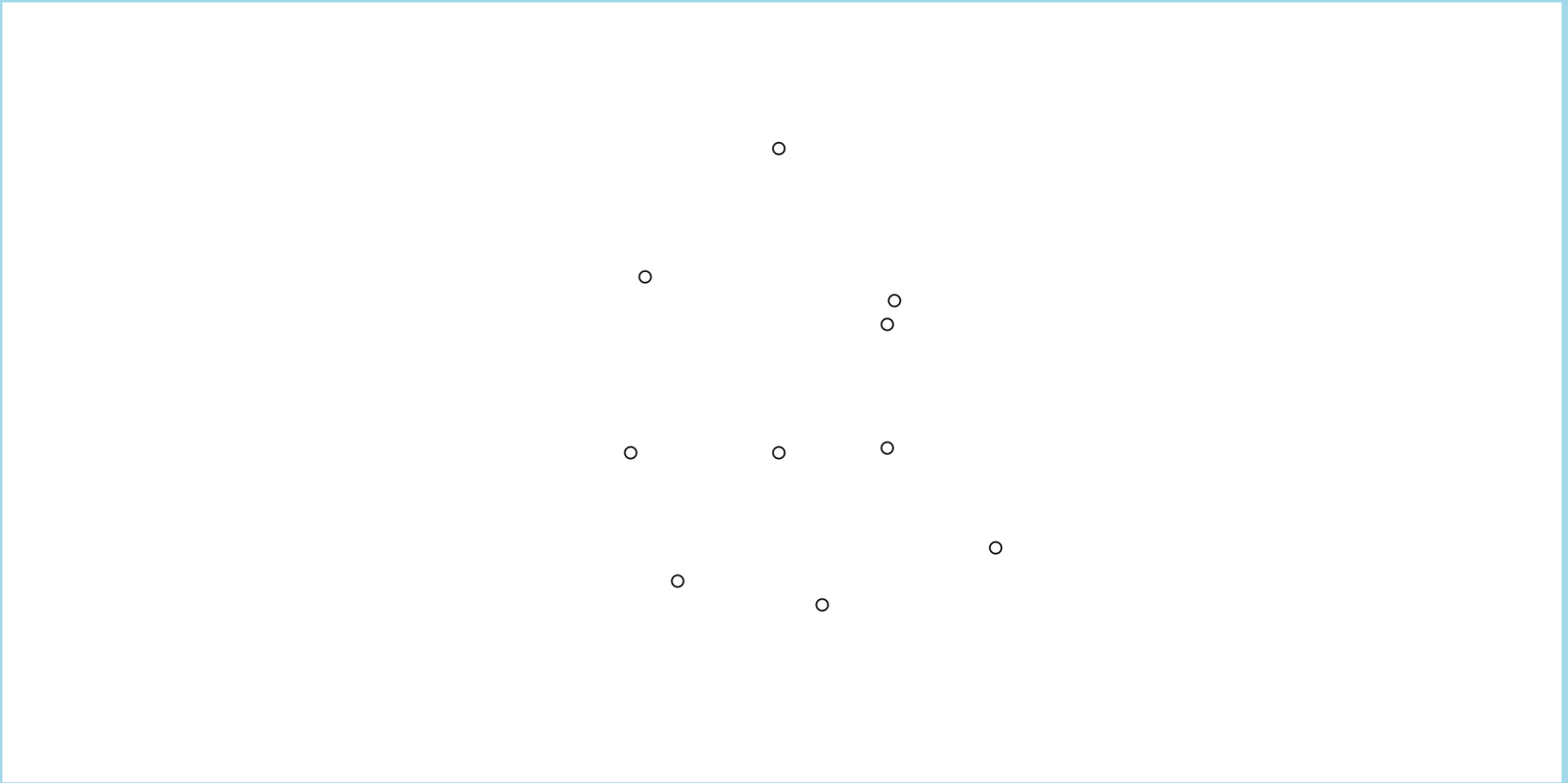
Points

```
1  library(sf)
2  proj <- st_crs('+proj=longlat +datum=WGS84')
3  long <- c(-116.7, -120.4, -116.7, -113.5, -115.5, -120.8, -119.5, -113.7, -
4  lat <- c(45.3, 42.6, 38.9, 42.1, 35.7, 38.9, 36.2, 39, 41.6, 36.9)
5  st_multipoint(cbind(long, lat)) %>% st_sfc(., crs = proj)
```

```
Geometry set for 1 feature
Geometry type: MULTIPOINT
Dimension:     XY
Bounding box:  xmin: -120.8 ymin: 35.7 xmax: -110.7 ymax: 45.3
Geodetic CRS:  +proj=longlat +datum=WGS84
```

Points

```
1 plot(st_multipoint(cbind(long, lat)) %>%
2                    st_sfc(., crs = proj))
```



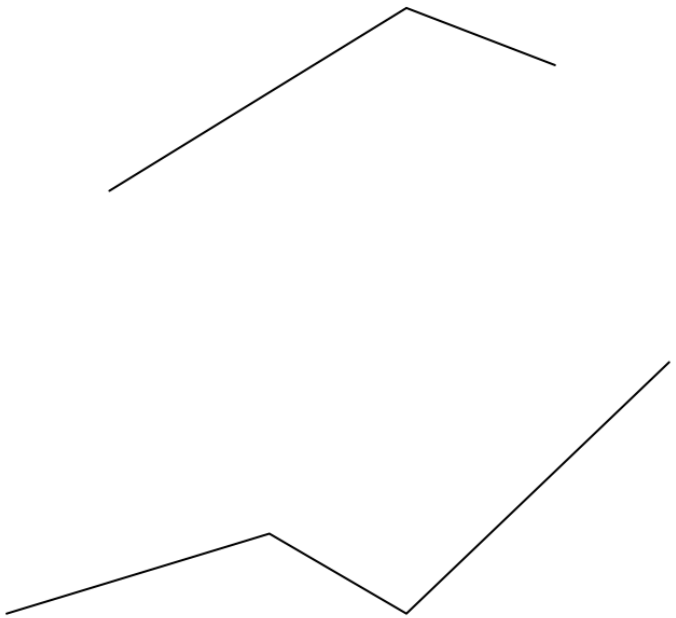
Lines

```
1 lon <- c(-116.8, -114.2, -112.9, -111.9, -114.2, -115.4, -117.7)
2 lat <- c(41.3, 42.9, 42.4, 39.8, 37.6, 38.3, 37.6)
3 lonlat <- cbind(lon, lat)
4 pts <- st_multipoint(lonlat)
5
6 sfline <- st_multilinestring(list(pts[1:3,], pts[4:7,]))
7 str(sfline)
```

```
List of 2
 $ : num [1:3, 1:2] -116.8 -114.2 -112.9 41.3 42.9 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "lon" "lat"
 $ : num [1:4, 1:2] -111.9 -114.2 -115.4 -117.7 39.8 ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr [1:2] "lon" "lat"
 - attr(*, "class")= chr [1:3] "XY" "MULTILINESTRING" "sfg"
```

Lines

```
1  plot(st_multilinestring(list(pts[1:3,], pts[4:7,])))
```

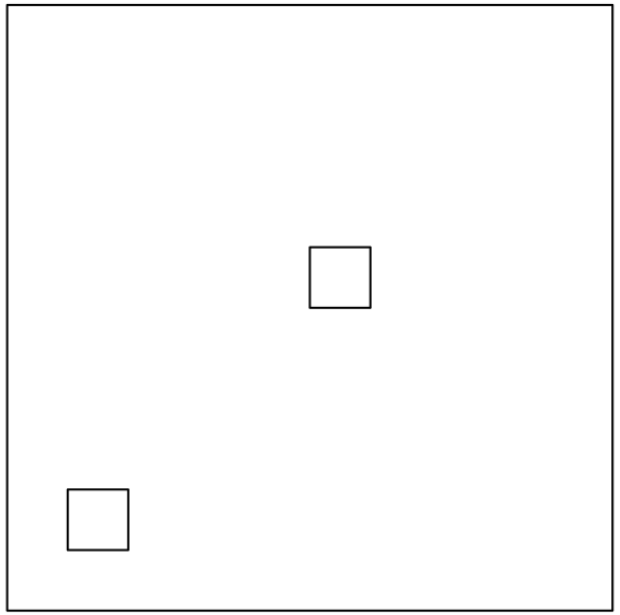


Polygons

```
outer = matrix(c(0,0,10,0,10,10,0,10,0,0),ncol=2, byrow=TRUE)
hole1 = matrix(c(1,1,1,2,2,2,2,1,1,1),ncol=2, byrow=TRUE)
hole2 = matrix(c(5,5,5,6,6,6,6,5,5,5),ncol=2, byrow=TRUE)
coords = list(outer, hole1, hole2)
pl1 = st_polygon(coords)
```

Polygons

```
1 plot(pl1)
```



But what about actual data?

Convert a data frame to `sf` object

- Useful for situations where point locations given as columns in spreadsheet
- Requires that you the projection used when the data were collected
- Using the `meuse` dataset (use `?sp::meuse` to learn more about it)

```
1 library(sp)
2 data(meuse)
3 head(meuse, n=3)[,1:10]
```

```
      x      y cadmium copper lead zinc  elev       dist   om ffreq
1 181072 333611    11.7     85  299 1022 7.909 0.00135803 13.6     1
2 181025 333558     8.6     81  277 1141 6.983 0.01222430 14.0     1
3 181165 333537     6.5     68  199  640 7.800 0.10302900 13.0     1
```

Convert a data frame to `sf` object

- Using the `x` and `y` columns in the data
- `agr` defines the attribute-geometry-relationship
- `constant`, `aggregate`, and `identity`

```
meuse_sf = st_as_sf(meuse, coords = c("x", "y"),
                    crs = 28992, agr =
                      "constant")
meuse_sf[1:2,1:10]
```

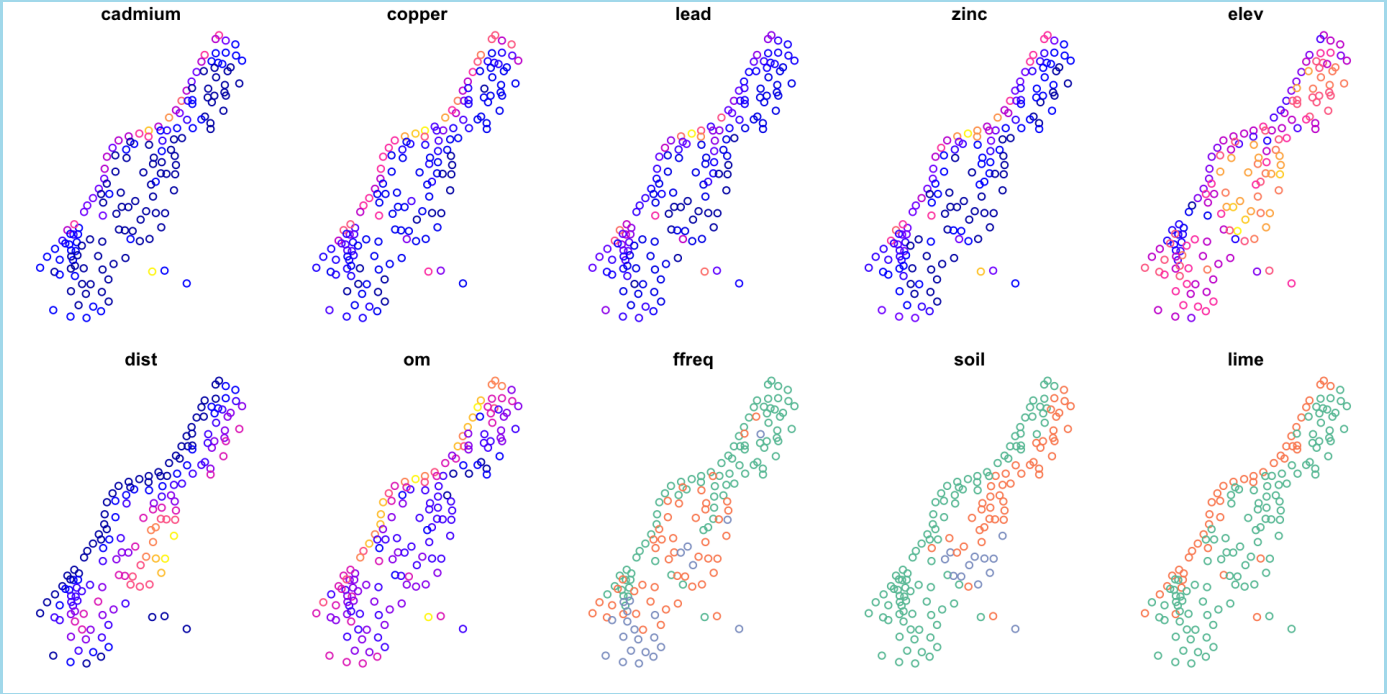
```
Simple feature collection with 2 features and 10 fields
Attribute-geometry relationships: constant (10)
Geometry type: POINT
Dimension:     XY
Bounding box:  xmin: 181025 ymin: 333558 xmax: 181072 ymax: 333611
Projected CRS: Amersfoort / RD New
  cadmium copper lead zinc  elev       dist   om ffreq soil lime
```

```
1    11.7     85  299 1022 7.909 0.00135803 13.6     1     1     1
2     8.6     81  277 1141 6.983 0.01222430 14.0     1     1     1
                geometry
1 POINT (181072 333611)
2 POINT (181025 333558)
```

Plotting `sf` objects

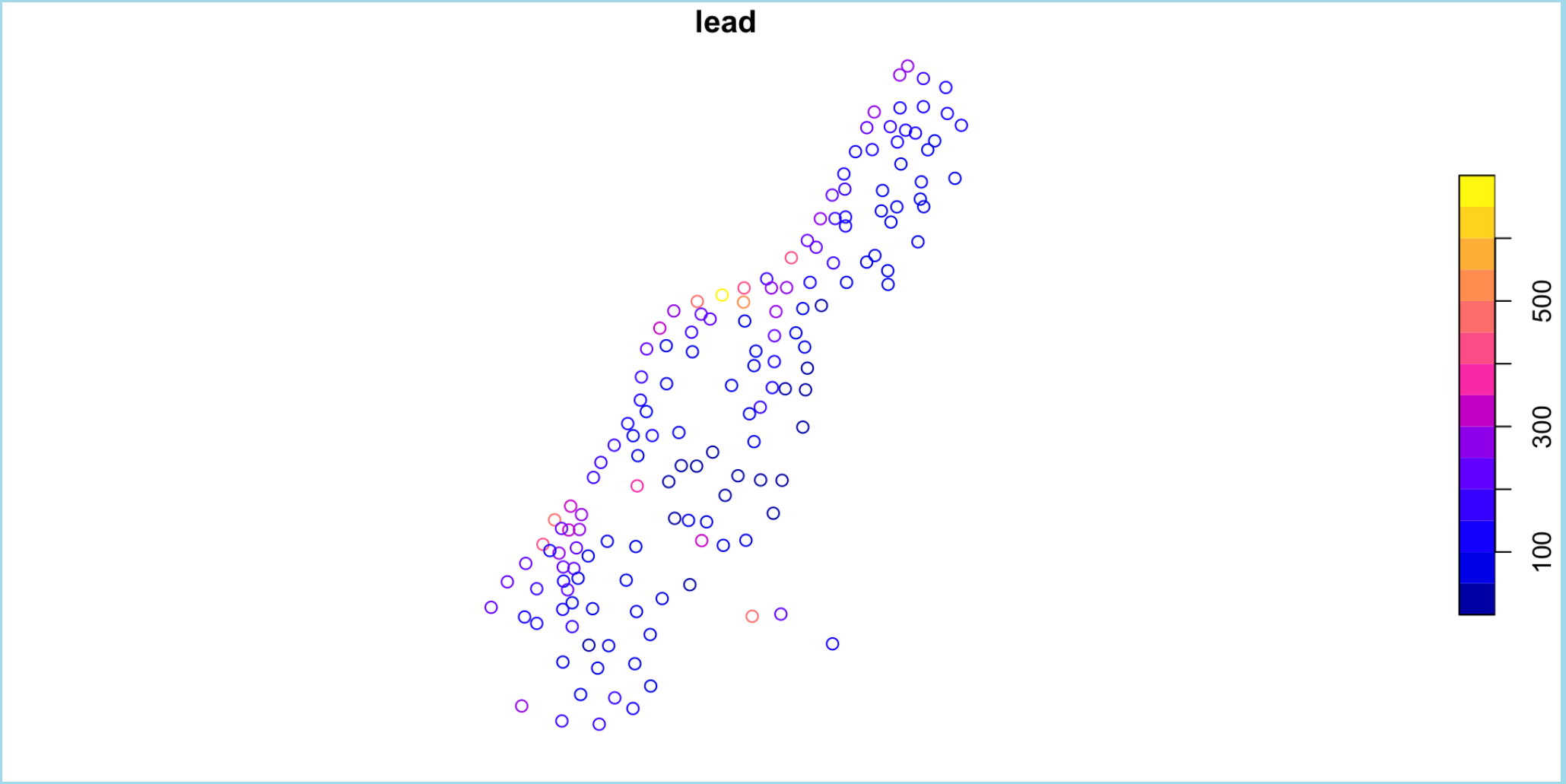
- Quick way to check your data
- Remember that `sf` has `geometry` and `attributes`

```
1  plot(meuse_sf)
```



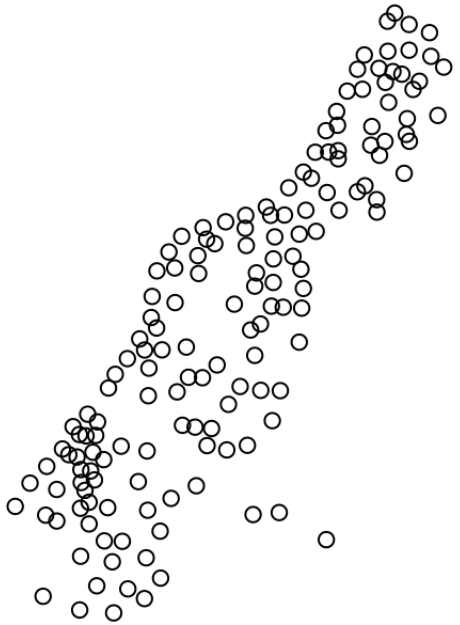
Plotting sf objects

```
1  plot(meuse_sf['lead'])
```



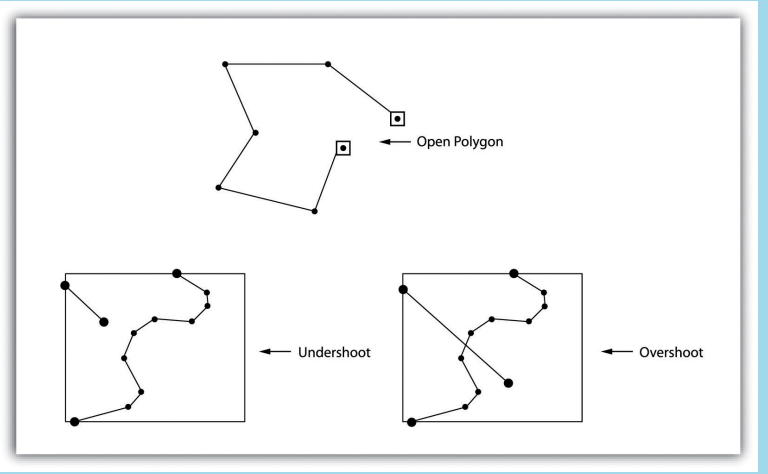
Plotting sf objects

```
1  plot(st_geometry(meuse_sf))
```



Common Problems with Vector Data

- Vectors and scale
- Slivers and overlaps
- Undershoots and overshoots
- Self-intersections and rings



Topology Errors - Saylor Acad.

We'll use `st_is_valid()` to check this, but fixing can be tricky

Fixing Problematic Topology

- `st_make_valid()` for simple cases
- `st_buffer` with `dist=0`
- More complex errors need more complex approaches

A Note on Vectors

Moving forward we will rely primarily on the `sf` package for vector manipulation. Some packages require objects to be a different class. `terra`, for example, relies on `SpatVectors`. You can use `as()` to coerce objects from one type to another (assuming a method exists). You can also explore other packages. Many packages provide access to the 'spatial' backbones of `R` (like `geos` and `gdal`), they just differ in how the "verbs" are specified. For `sf` operations the `st_` prefix is typical. For `rgeos` operations, the `g` prefix is common.