Spatial data with

Spatial *is* special

- Complex: geometry and attributes
- The earth is not flat (coordinate reference systems)
- Size: lots and lots of it, multivariate, time series
- Special plots: maps
- First Law of Geography: nearby things are similar
  - Statistical assumptions: violated
  - Interpolation: possible

Don't we have GIS for that?

GIS – R*

- Visual interaction — Data & model focused
- Data management — Analysis
- Geometric operations — Attributes as important
- Standard workflows — Creativity & innovation
- Single map production — Many (simpler) maps
- Click, click, click & click — Repeatability (single script)
- Speed of execution — Speed of development
- Cumbersome — Easy & powerful (& free)

* there are many different GISs and they evolve

Spatial is not ***that*** special

- Spatial data is just an extension of “base” data --- the same ***data science*** principles apply

- Collection, organization, cleaning, exploring, summarizing, analyzing, modeling, predicting, storage, big data...

Representation of space

Objects

Discrete entities, defined by coordinates

points, lines, areas

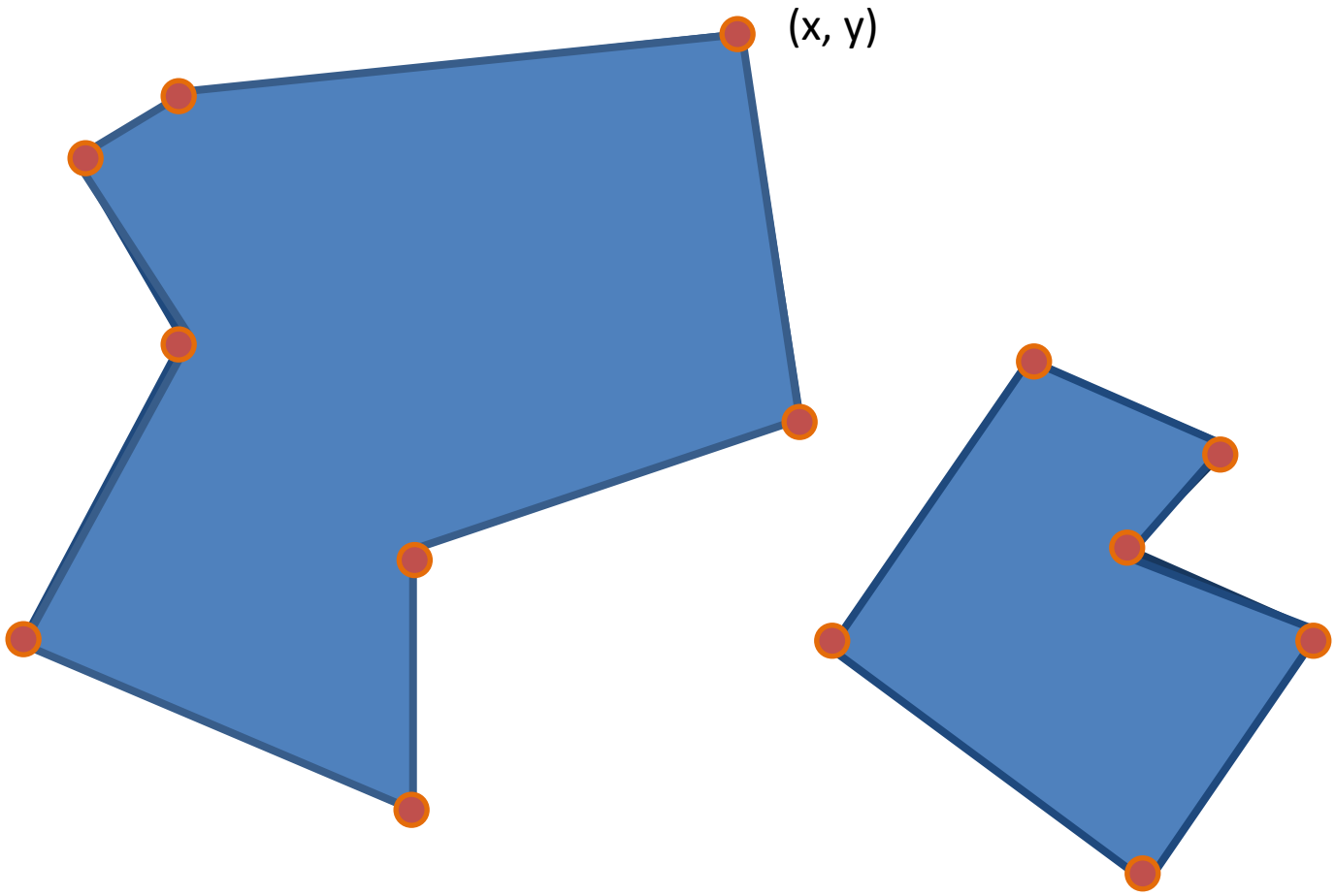
typically represented as *vector* data

Fields

Continuously varying properties

typically represented as *raster* data

Vector data



The '*terra*' package defines *classes* to represent spatial data.

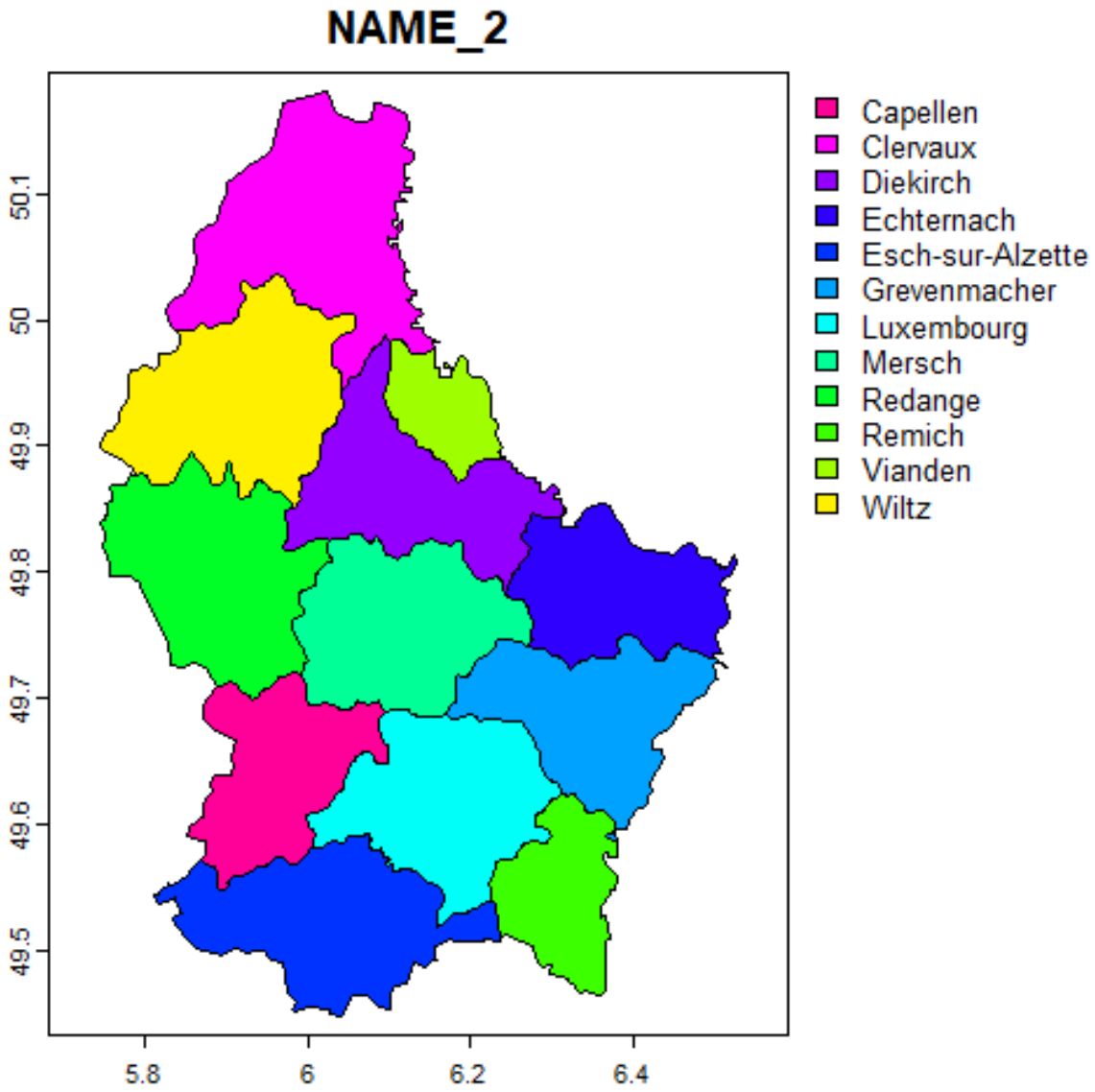
```
> library(terra)
# terra 1.6.48

xy <- cbind(c(0,1), c(0,1))
v  <- vect(xy, crs="+proj=longlat")
v
# class       : SpatVector
# geometry    : points
# dimensions  : 2, 0  (geometries, attributes)
# extent      : 0, 1, 0, 1  (xmin, xmax, ymin, ymax)
# coord. ref. : +proj=longlat +datum=WGS84 +no_defs
```

```
> v <- vect("lux.shp")
> v

# class       : SpatVector
# geometry    : polygons
# dimensions  : 12, 6  (geometries, attributes)
# extent      : 5.74, 6.53, 49.45, 50.18 (xmin, xmax, ymin, ymax)
# source      : lux.shp
# coord. ref. : lon/lat WGS 84 (EPSG:4326)
# names       :  ID_1   NAME_1  ID_2   NAME_2  AREA   POP
# type        : <num>    <chr> <num>    <chr> <num> <int>
# values      :     1 Diekirch     1 Clervaux   312 18081
#                   1 Diekirch     2 Diekirch   218 32543
#                   1 Diekirch     3  Redange   259 18664
```

```
> plot(v, "NAME_2")
```


NAME_2
Capellen
Clervaux
Diekirch
Echternach
Esch-sur-Alzette
Grevenmacher
Luxembourg
Mersch
Redange
Remich
Vianden
Wiltz
50.1
50
49.9
49.8
49.7
49.6
49.5
5.8
6
6.2
6.4

```
> values(v)
```

	ID_1	NAME_1	ID_2	NAME_2	AREA
1	1	Diekirch	1	Clervaux	312
2	1	Diekirch	2	Diekirch	218
3	1	Diekirch	3	Redange	259
4	1	Diekirch	4	Vianden	76
5	1	Diekirch	5	Wiltz	263
6	2	Grevenmacher	6	Echternach	188
7	2	Grevenmacher	7	Remich	129
8	2	Grevenmacher	12	Grevenmacher	210
9	3	Luxembourg	8	Capellen	185
10	3	Luxembourg	9	Esch-sur-Alzette	251
11	3	Luxembourg	10	Luxembourg	237
12	3	Luxembourg	11	Mersch	233

example methods

see `?terra` for more

erase or "-"	erase parts of a SpatVector
intersect or "*"	intersect SpatVectors
union or "+"	union SpatVectors
cover	update and identity a SpatVector with another one
symdif	symmetrical difference of two SpatVectors
aggregate	combine polygons into larger ones
disagg	explode: turn polygon parts into separate polygons
crop	clip a SpatVector object using a rectangle (Extent object)
select	select - interactively select spatial features
click	identify attributes by clicking on a map
merge	Join table (in the sp package)
extract	spatial queries between SpatVector and SpatRaster objects
as.data.frame	coerce into a data.frame