

H3 as a GeoParquet pyramid builder

Scale-aware geo-analytics

Guillaume Sueur, Geomermaids, H3 Conference, Uber NYC, 23 September 2026



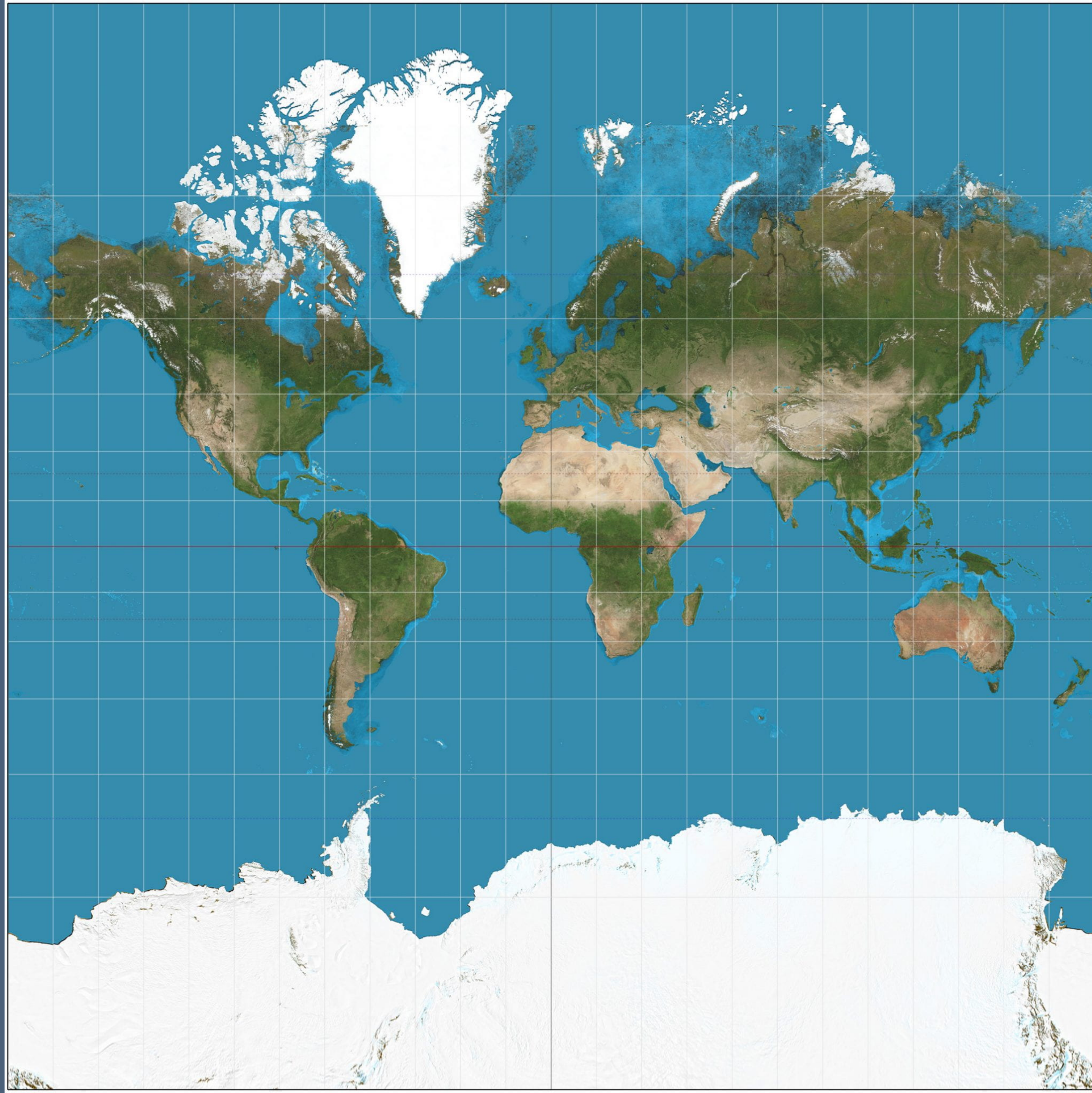
Why is Geospatial so special ?

Or why can't we fit into other industries solutions ?

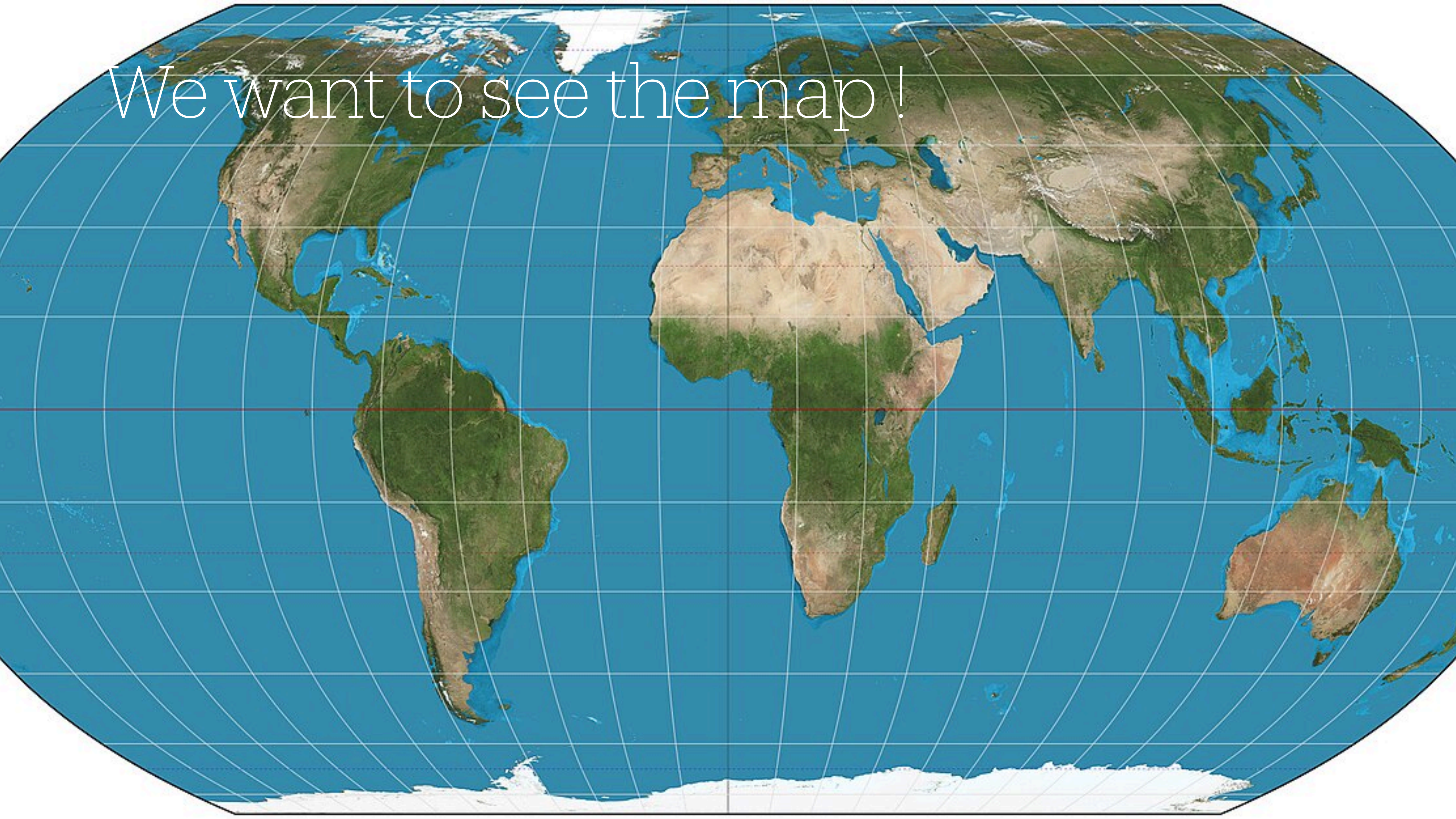
2 main reasons (to my knowledge)...

1. We want to
see the map !

No, not this map anymore...



We want to see the map!



Imagine doing
this with an
Excel file...

Like a psychopath...

Formatted Corona Virus Data

Search

File Home Insert Formulas Data Review View Help

Font: A A, Bold, Italic, Underline, Text Color, Background Color, Wrap Text, Merge & Center, Alignment, Number, Styles, Cells

Conditional Formatting, Format as Table, Cell Styles, Insert, Delete, Format, AutoSum, Fill, Clear

Warning: If you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format. Don't show again Save As...

	E	F	G	H	I	J	K	L	M	N	O	P	Q
	total_cases	new_cases	total_deaths	new_deaths	total_cases	new_cases	total_deaths	new_deaths	total_tests	new_tests	total_tests	new_tests	tests_units
	1795	80	65	5	39.716	1.77	1.438	0.111	14850	1520	0.329	0.034	tests performed
	1894	99	79	14	41.907	2.19	1.748	0.31	16379	1529	0.362	0.034	tests performed
	1975	81	82	3	43.699	1.792	1.814	0.066	18027	1648	0.399	0.036	tests performed
	2272	69	98	3	50.27	1.527	2.168	0.066	22805	3047	0.505	0.067	tests performed
	2432	160	105	7	53.81	3.54	2.323	0.155	24374	1569	0.539	0.035	tests performed
	2432	0	109	4	53.81	0	2.412	0.089	26457	2083	0.585	0.046	tests performed
	2560	128	115	6	56.642	2.832	2.544	0.133	28650	2193	0.634	0.049	tests performed
	2658	98	122	7	58.811	2.168	2.699	0.155	30942	2292	0.685	0.051	tests performed
	2828	170	132	10	62.572	3.761	2.921	0.221	32712	1770	0.724	0.039	tests performed
	2930	102	134	2	64.829	2.257	2.965	0.044	34568	1856	0.765	0.041	tests performed
	3020	90	142	8	66.82	1.991	3.142	0.177	36611	2043	0.81	0.045	tests performed
	3132	112	151	9	69.299	2.478	3.341	0.199	39228	2617	0.868	0.058	tests performed
	3276	144	159	8	72.485	3.186	3.518	0.177	41786	2558	0.925	0.057	tests performed
	3423	147	165	6	75.737	3.253	3.651	0.133	44654	2868	0.988	0.063	tests performed
	3423	0	167	2	75.737	0	3.695	0.044	47406	2752	1.049	0.061	tests performed
	3767	344	185	18	83.348	7.611	4.093	0.398	49905	2499	1.104	0.055	tests performed
	3767	0	186	1	83.348	0	4.115	0.022	51900	1995	1.148	0.044	tests performed
	3990	223	197	11	88.283	4.934	4.359	0.243	53600	1700	1.186	0.038	tests performed
	4114	124	207	10	91.026	2.744	4.58	0.221	56058	2458	1.24	0.054	tests performed
	5744	57	36	2	225.256	2.235	1.412	0.078	302663	5509	11.869	0.216	tests performed
	5844	100	42	6	229.178	3.922	1.647	0.235	310700	8037	12.184	0.315	tests performed
	5956	112	45	3	233.57	4.392	1.765	0.118	319368	8668	12.524	0.34	tests performed
	6052	96	50	5	237.334	3.765	1.961	0.196	330134	10766	12.946	0.422	tests performed
	6152	100	52	2	241.256	3.922	2.039	0.078	338346	8212	13.269	0.322	tests performed
	6238	86	54	2	244.629	3.373	2.118	0.078	347847	9501	13.641	0.373	tests performed
	6289	51	57	3	246.629	2	2.235	0.118	353941	6094	13.88	0.239	tests performed
	6322	33	61	4	247.923	1.294	2.392	0.157	362136	8195	14.201	0.321	tests performed
	6366	44	61	0	249.648	1.725	2.392	0	366493	4357	14.372	0.171	tests performed
	6416	50	61	0	251.609	1.961	2.392	0	371377	4884	14.564	0.192	tests performed
	6458	42	63	2	253.256	1.647	2.471	0.078	380003	8626	14.902	0.338	tests performed
	6497	39	63	0	254.786	1.529	2.471	0	391530	11527	15.354	0.452	tests performed
	6533	36	67	4	256.197	1.412	2.627	0.157	406510	14980	15.942	0.587	tests performed
	6586	53	69	2	258.276	2.078	2.706	0.078	420996	14486	16.51	0.568	tests performed
	6612	26	70	1	259.295	1.02	2.745	0.039	431734	10738	16.931	0.421	tests performed

Average: 65314.83791 Count: 12 Sum: 718463.217

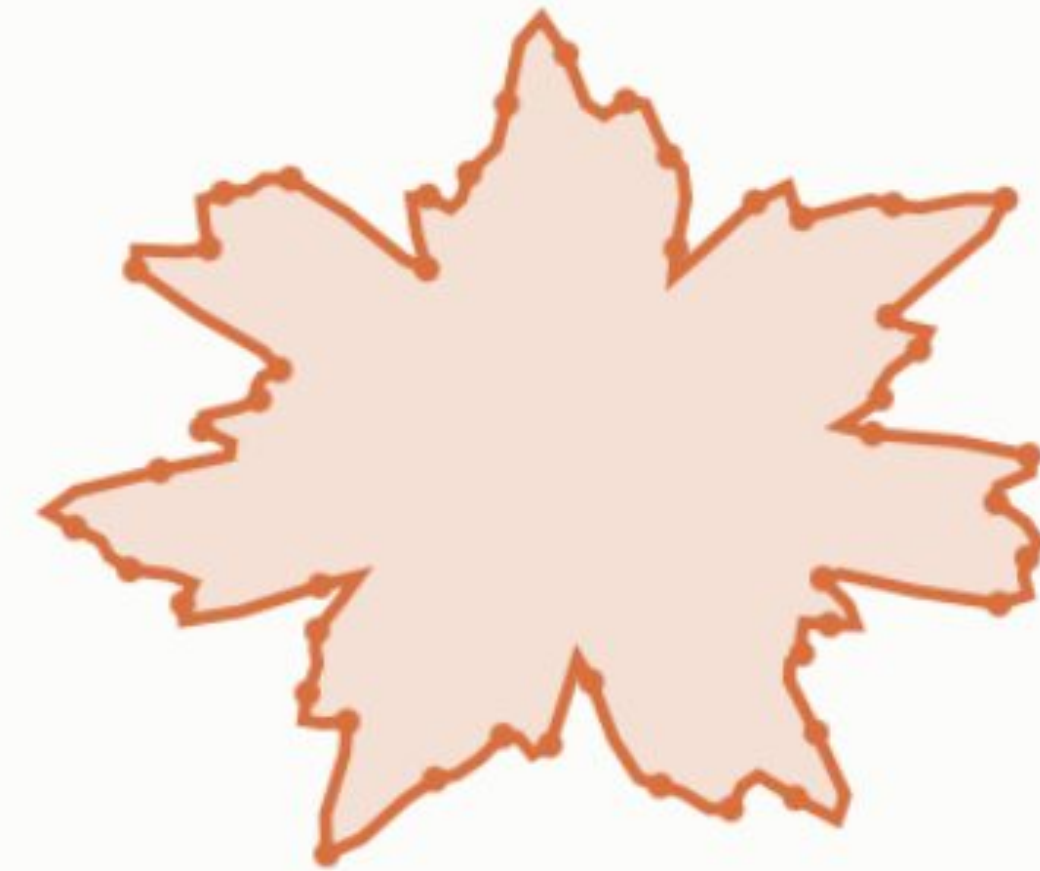
We don't know what's inside the box



what a row can be



4 vertices, 100 bytes

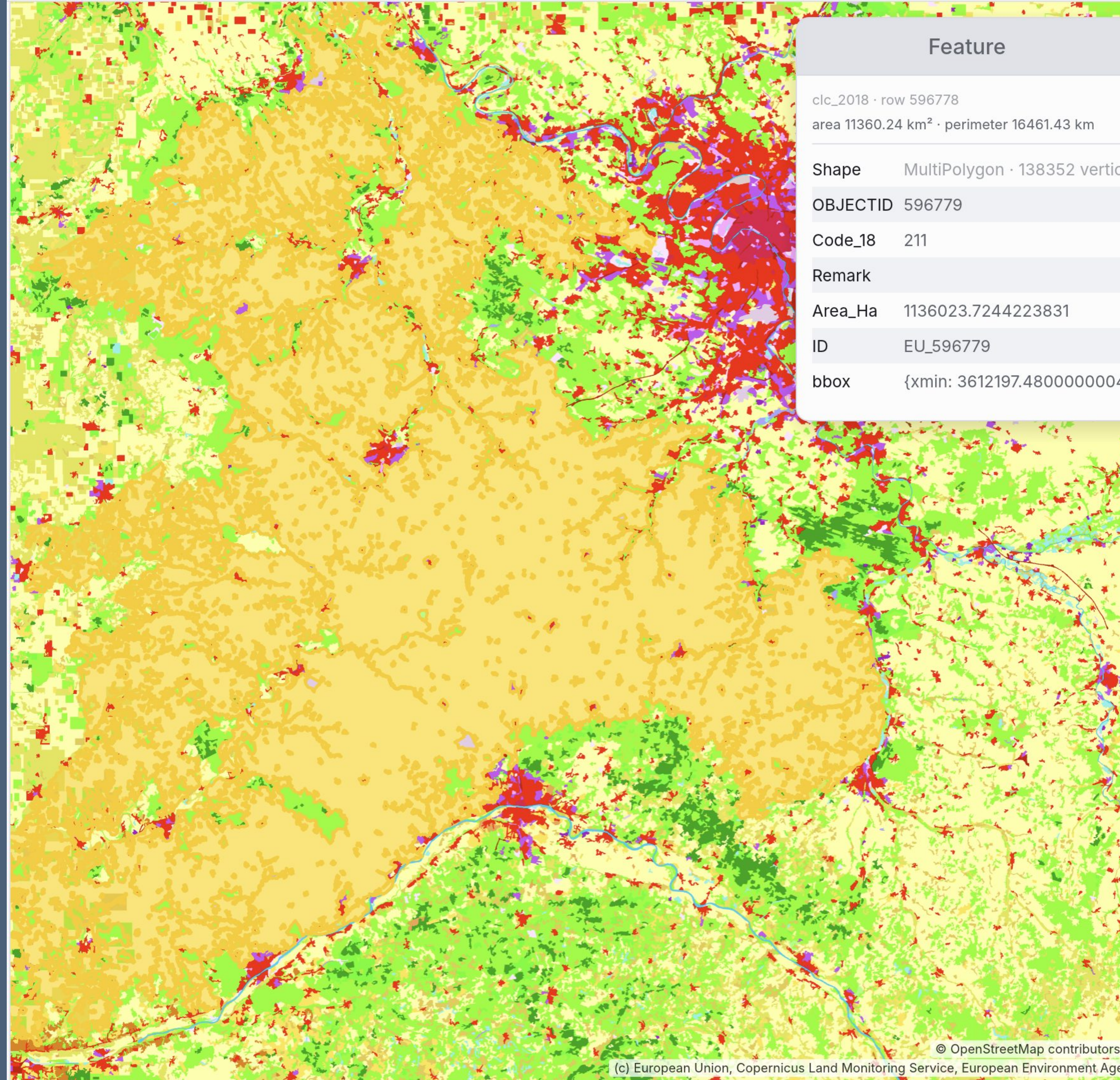


300,000 vertices, 5 MB

same column, same file, same 'SELECT geometry'

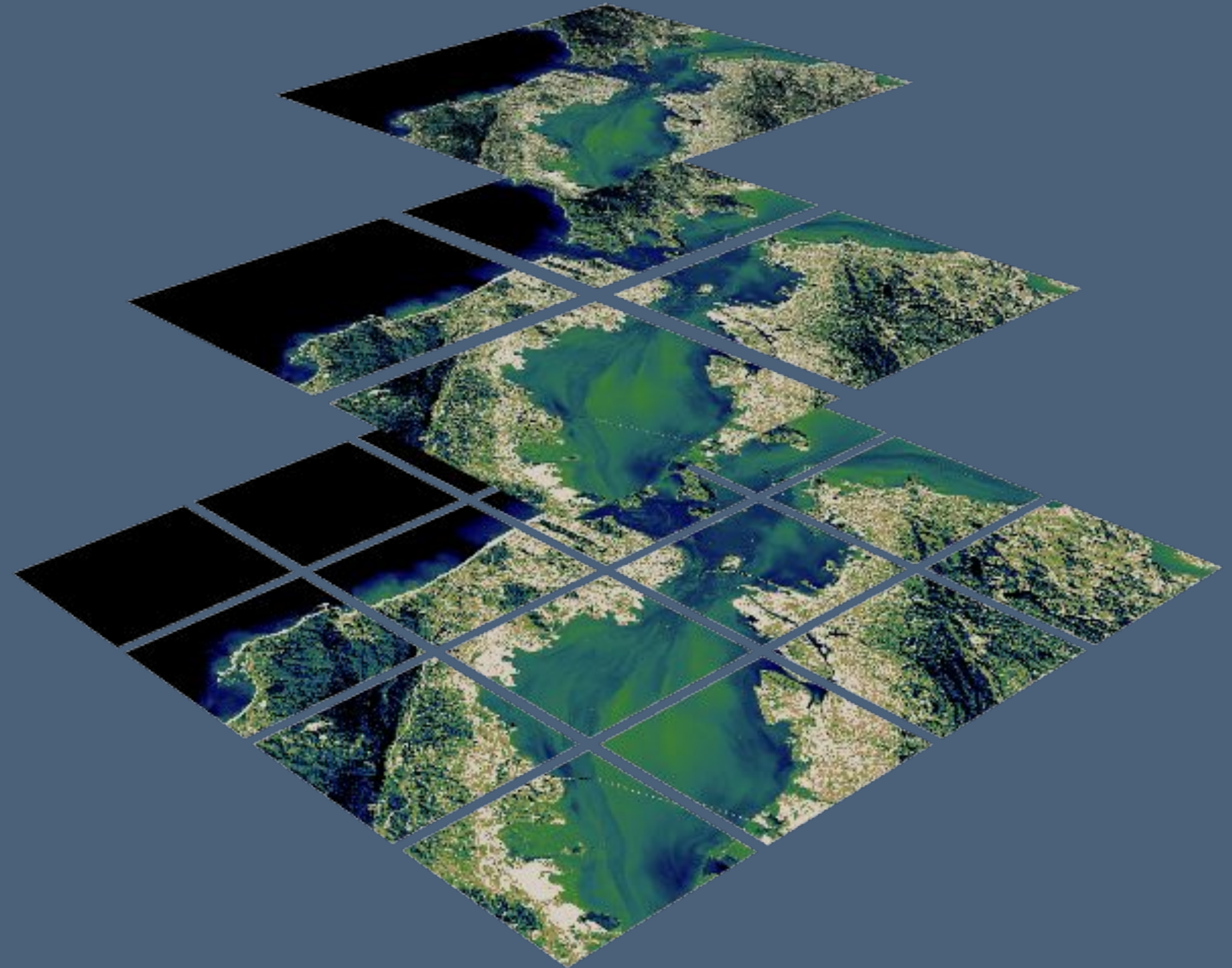
After more than 60 years of GIS...

- One actual geometry column type
- Some subtypes more to help with geometric operations than to advertise about the content
- No SMALLGEOM vs BIGGEOM
- No GEOM(200)



Raster data has an easier path

- same-sized pixels
- Same-sized content per pixel (1, 2, 3, maybe some more bands)
- Easy to resample
- Easy to store along with the original data



Geoparquet in short

Parquet, plus a geometry column, plus metadata

- **Parquet** is the Apache columnar file format of the analytics world : one file, columns stored separately, compressed, with a nice footer describing what's in there (schema, row groups, per column stats)
- **GeoParquet** adds one column of geometry (usually WKB blobs), a CRS, and bbox metadata at row-group level
- Supports HTTP-range requests making it a great choice for Cloud Native Geospatial
- This is the format you want under your geo-analytics. The rest of this talk is about the one thing it is bad at.

Geoparquet caveats

Or what makes it a format hard to handle

- No per-row spatial index, so you rely on row groups indexes and may download way more content than you need.
- Obviously no overviews or LOD (level of detail) concept in the specification even if some discussions are ongoing
- No way to prevent “too big to draw” objects from being accessed

Options to make overviews...

A second geometry column

- Easy to generate, Row count untouched, indexes are the same
- Still queryable for analytics
- Heavier file
- Which column will a reader choose, and when ? Needs some added content in metadata...

Options to make overviews...

A sidecar file

- Easy to generate, Row count untouched, indexes are the same
- Still queryable for analytics
- 2 files to keep in sync, but the second one is not mandatory
- A reader may not implement its reading, making it more versatile

Options to make overviews...

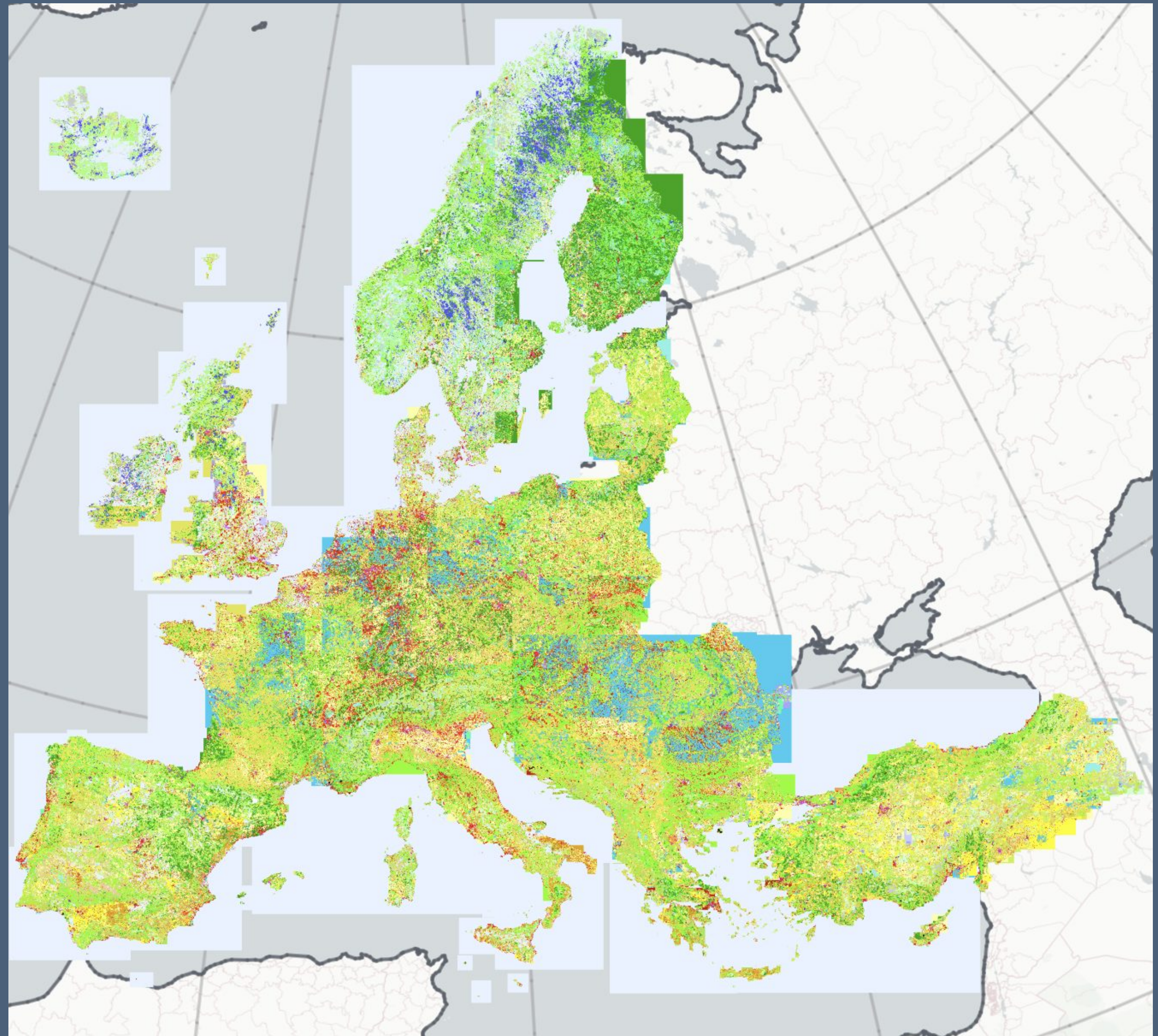
Use bounding boxes

- Easy to generate, even part of GeoParquet v1.1 spec and probably v.2.0
- Still queryable for analytics
- Super efficient for displaying complex geometries at small scale
- May look awkward at bigger scale

Options to make overviews...

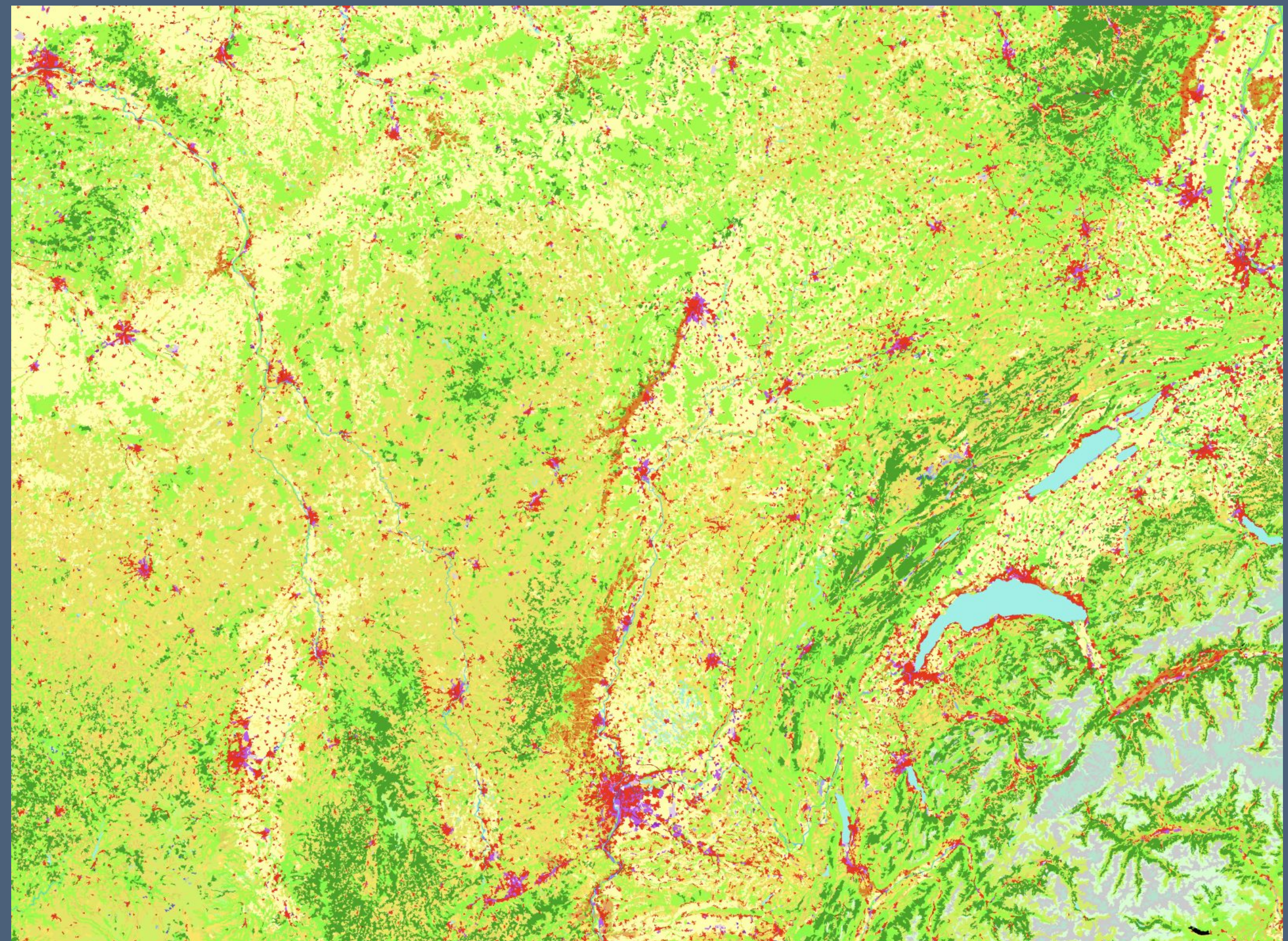
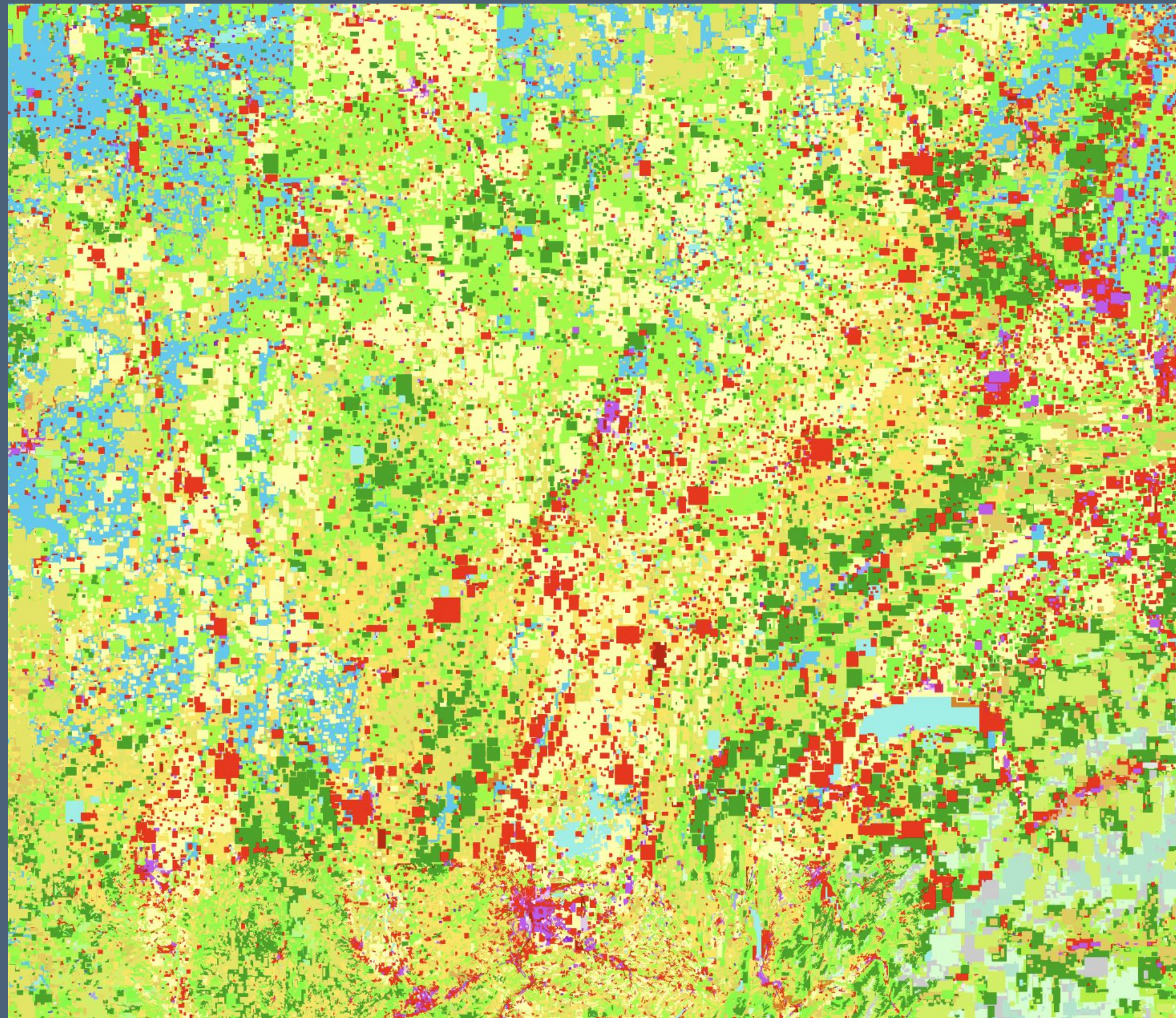
Use bounding boxes

- 2.5M features here
- Displayed in a few seconds



Options to make overviews...

Use bounding boxes

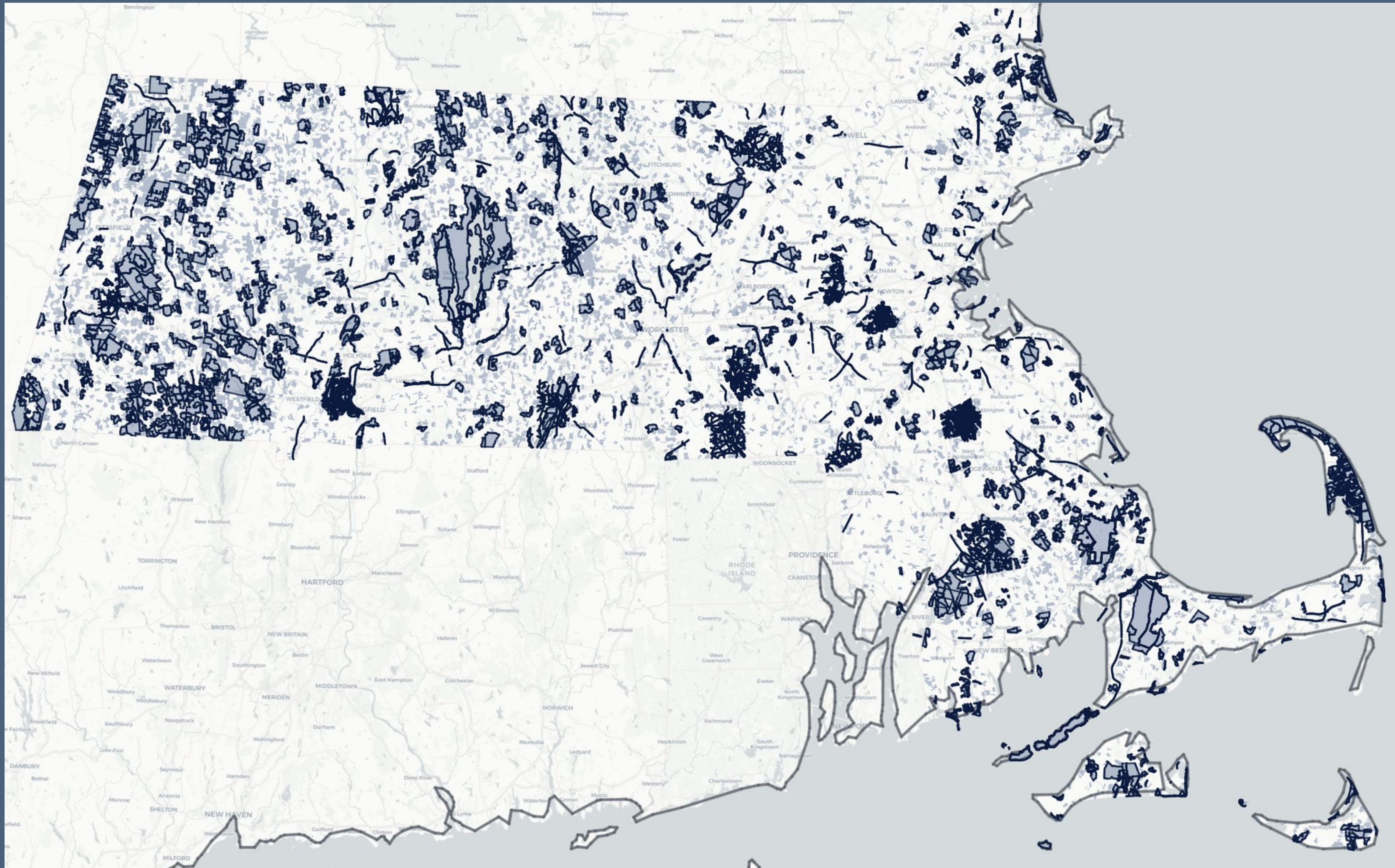


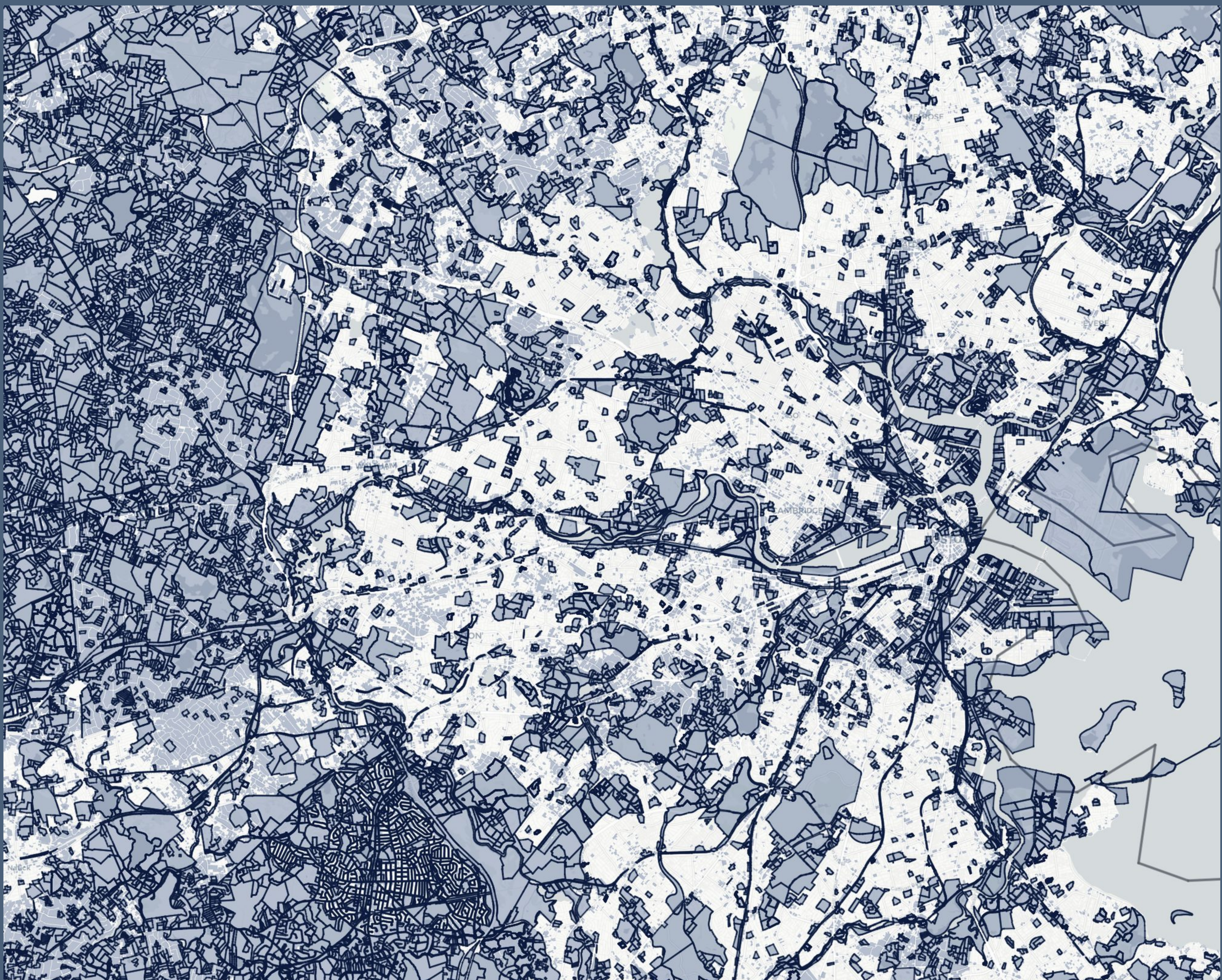
Options to make overviews...

LOD inside one file (COGP)

- A COGP file is a valid GeoParquet 1.1 file whose row groups are physically ordered from coarse to fine rendering detail, with file-level metadata describing where each level ends.
- COGP is feature-level: it reorders features across row groups; it does not simplify, aggregate, or duplicate them. Each source feature appears in exactly one row group, with its geometry preserved verbatim.
- A COGP-aware reader can stream just the leading row groups needed for its target rendering resolution
- Requires specific implementation (may need an extension of the spec)
- May look awkward too because you over-represent coarse areas and undersample dense areas

cf <https://github.com/Kanahiro/cloud-optimized-geoparquet>





Why not use H3-based pyramidal structure ?

- GeoParquet supports partitioning over H3
- H3 provides a fast, easy-to-compute spatial index

One file per cell, one folder per resolution

That's the whole idea

parcels/

h3-pyramid.json

collection.json

r3/832a06fffffffff.parquet ... 7 files

r4/842a069fffffffff.parquet ... 27 files

r5/852a0693fffffffff.parquet ... 123 files

r6/862a06907fffffffff.parquet ... 697 files

r6/___HIVE_DEFAULT_PARTITION___.parquet

descriptor: levels, method, cell lists, px per cell

STAC, optional, for other tools

overview, dissolve, 27 rows

overview, dissolve, 123 rows

overview, dissolve, 697 rows

leaf: the 2,557,398 source parcels, verbatim

the one row with no geometry

file name = H3 cell id. Level = resolution. Nothing to list, nothing to look up:

a reader derives the file it needs from a coordinate and a scale.

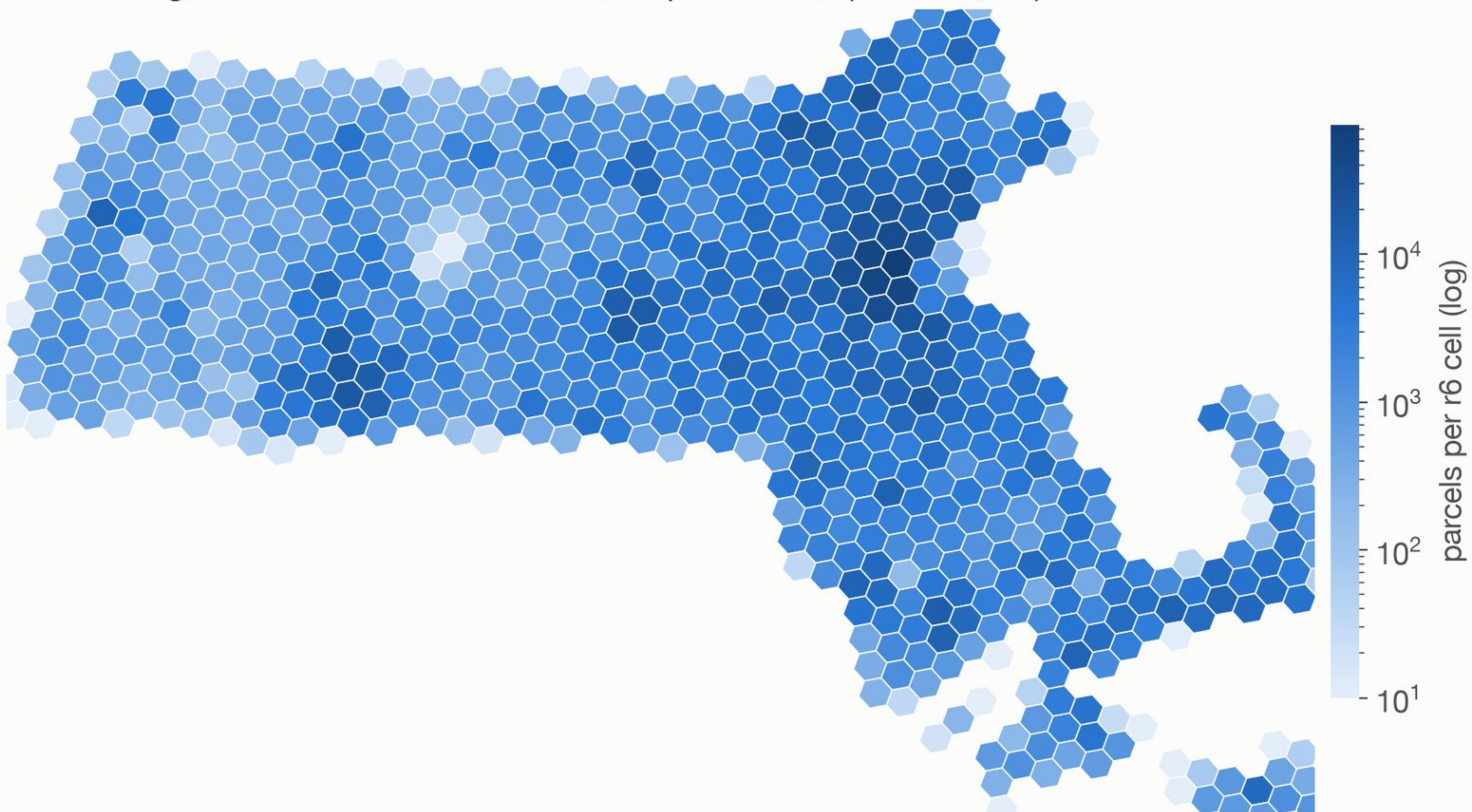
- The file name is the spatial index
- Every file is a plain, valid GeoParquet file

The ground floor is just your data

Resolution 6, —> 697 files

- 2.5M Massachusetts parcels, geometries untouched
- One file per H3 cell, Hilbert-sorted inside
- A cell with too many rows ? It splits into its children
- Nothing clever yet, this is just decent partitioning

The leaf tiling: 697 r6 cells hold data, 1 to 75,352 parcels each (median 1,874)



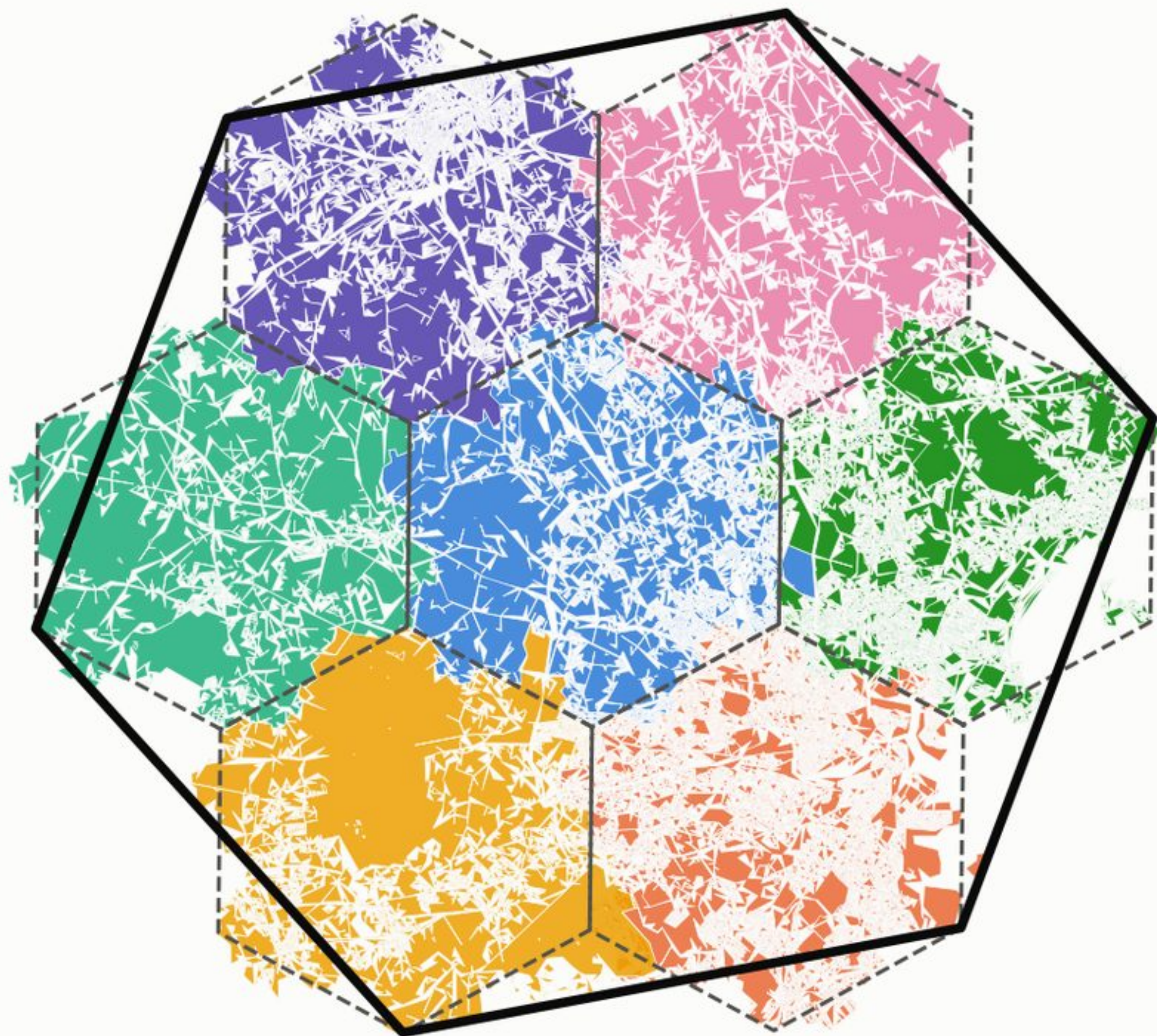
Reference resolution r6 was preselected by the density table (largest cell 75,352 < 250,000 target), so no cell was split into r7 children here; the densest is 862a30667fffff (Boston).

Going up : one file, one cell, seven rows

Seven-ish. It's H3, you know why...

- One row per child cell
- Its geometry : the union of everything inside that child
- Its columns : count, area_sum
- Built bottom-up, each level from the one below

One overview file, r4/842a307fffffffff.parquet: 7 rows, one per r5 child, each the dissolve of that child's parcels



row	child cell (r5)	parcels	parts
1	852a3063fffffffff	77,511	2,663
2	852a3067fffffffff	303,160	8,460
3	852a306bfffffffff	33,147	672
4	852a306ffffffffff	64,484	1,732
5	852a3073fffffffff	52,671	1,463
6	852a3077fffffffff	143,333	4,936
7	852a307bfffffffff	67,452	2,428

solid: the r4 cell = the file
dashed: its 7 r5 children = the rows
'parcels' is the dissolve's own count column

Rows are grouped by the prefix of the leaf file they came from, so a child's parcels sit in exactly one parent file even where the child spills over the parent's edge.

And it is still a table

Scale-aware geo-analytics

- Same answer from r3, with 27 rows
- Counts and areas are exact at every level, only the shapes get approximate
- Need the real parcels of one cell ? The file name is the WHERE clause

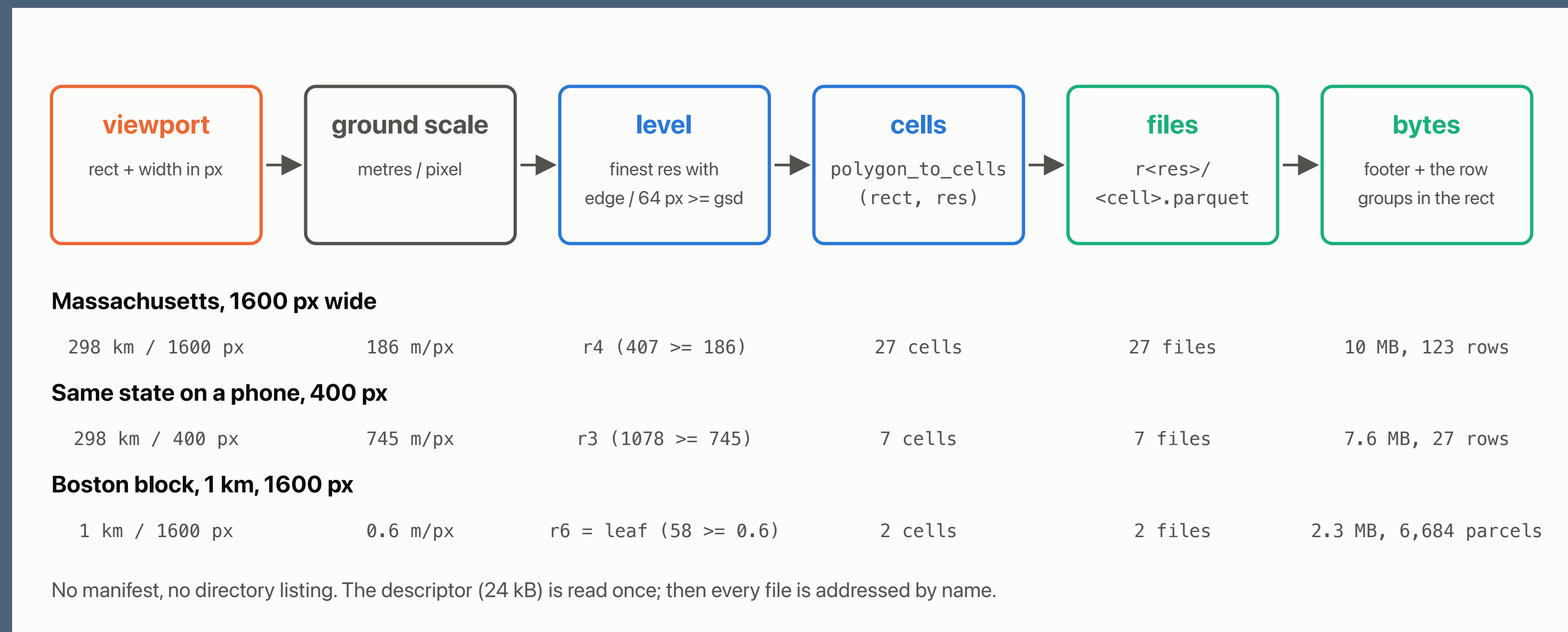
```
SELECT sum("count") FROM 'r4/*.parquet';
```

```
2,557,398 parcels      123 rows read      5 ms
```

How does a reader find its way ?

Arithmetic, not an index

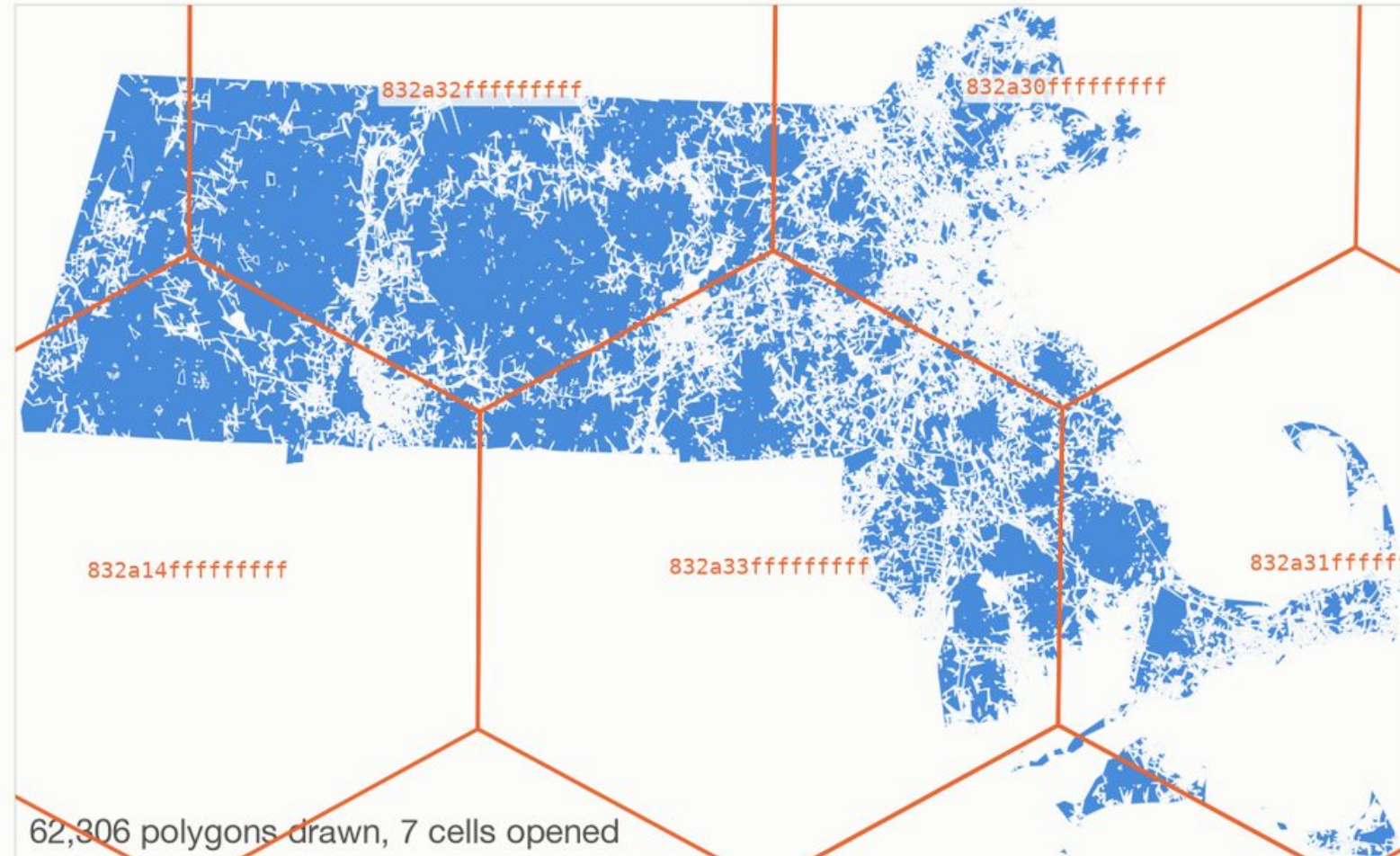
- Viewport gives meters per pixel
- Pick the resolution where one cell is about 64 pixels wide
- `polygon_to_cells(viewport, res)` gives the file names
- One level at a time, re-planned on zoom like tiles do
- One 24 kB JSON says which cells exist, for 855 files



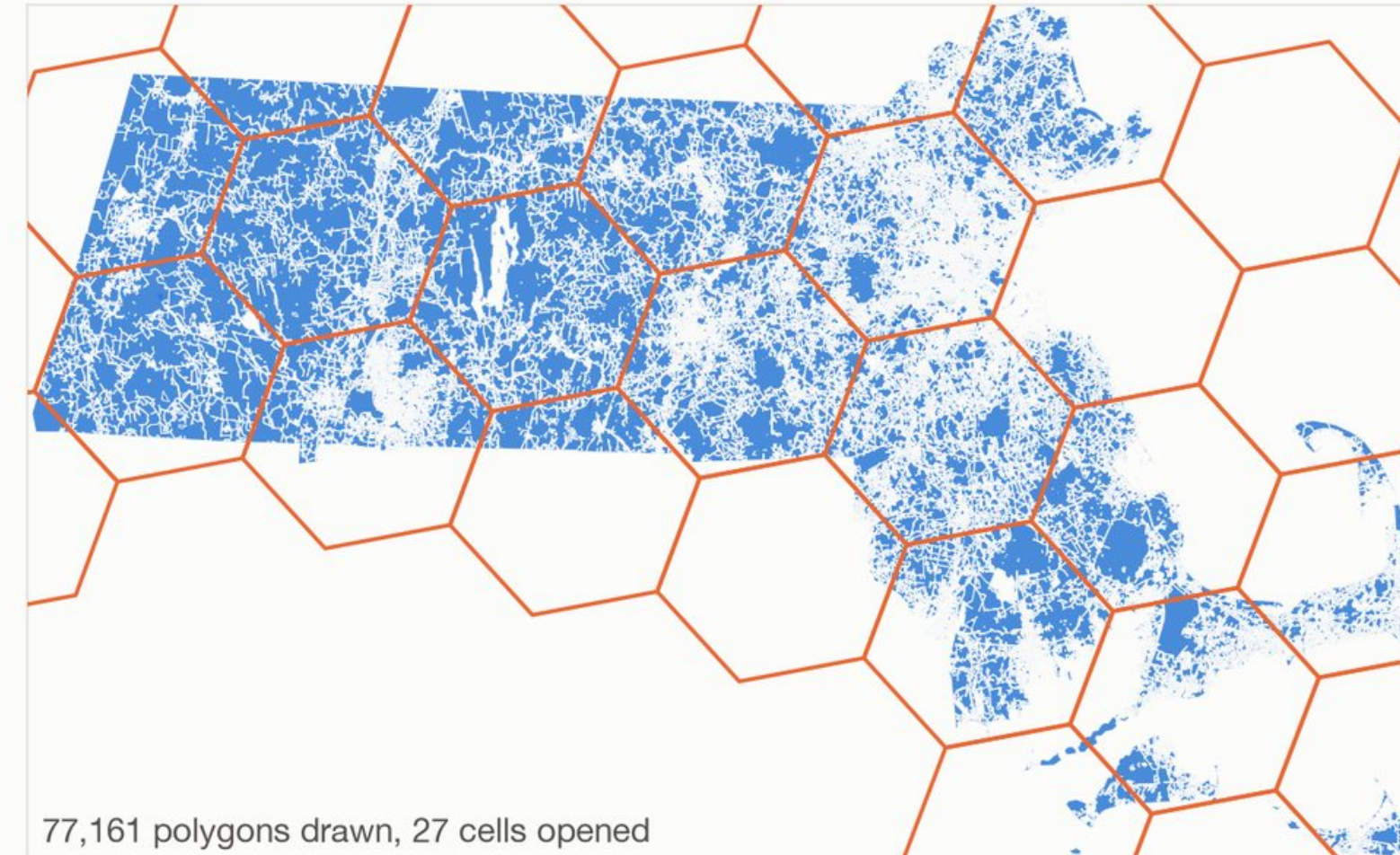
Same state, four levels

One pyramid, four levels: the rows read (blue) and the cells that name the files (orange)

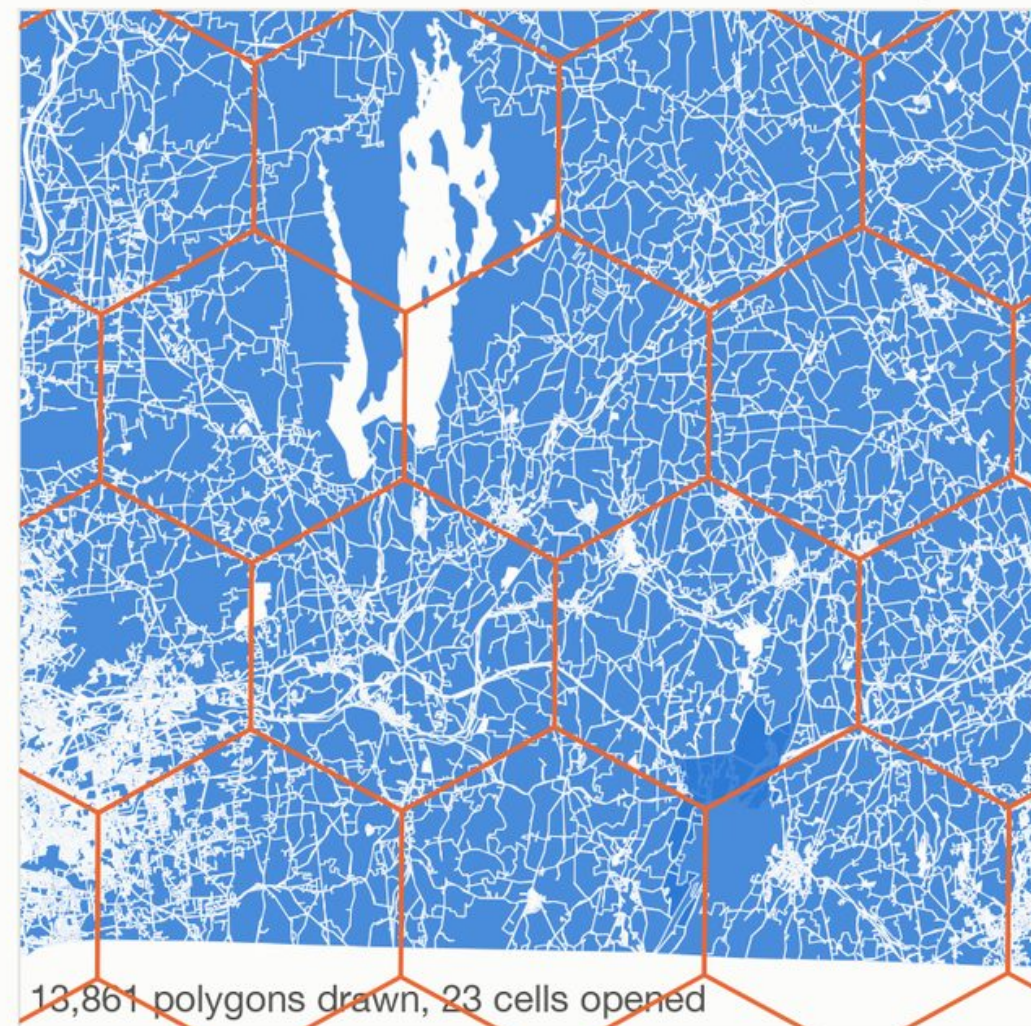
r3: 7 cells = 7 files, 27 rows, 7.6 MB (state on a phone)



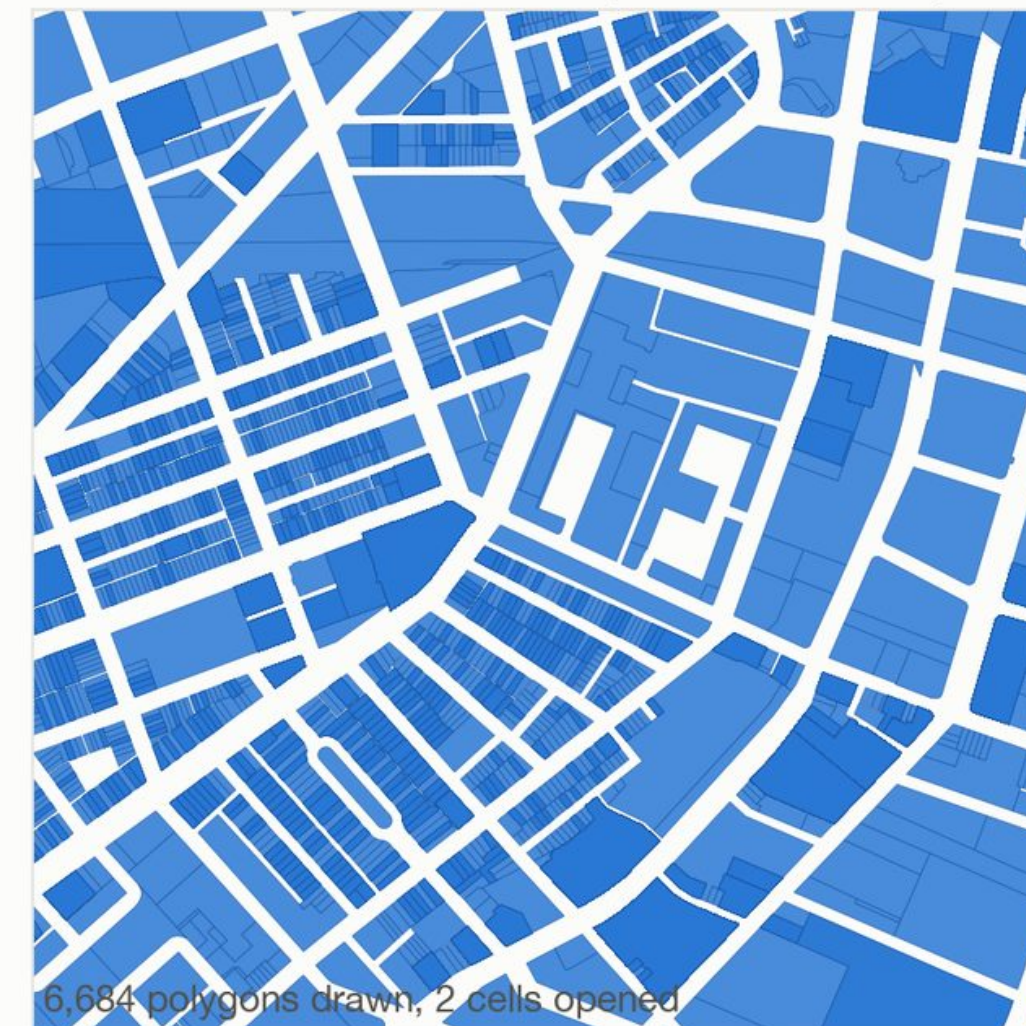
r4: 27 cells = 27 files, 123 rows, 10.3 MB (state at 1600 px)



r5: 23 cells = 23 files, 112 rows, 2.6 MB (county on a phone)



r6 leaf: 2 cells = 2 files, 6,684 parcels, 2.3 MB (Boston block)



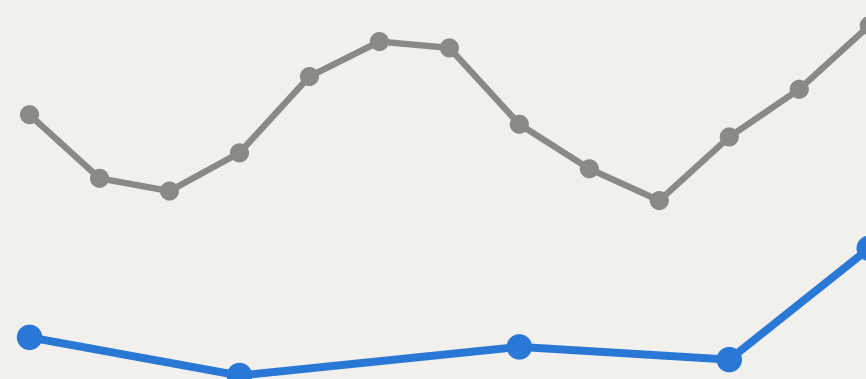
Three ways to climb

the method is written in the metadata

- Simplify : same rows, fewer vertices. Fine for lines or if you want to keep all features
- Prune : keep the biggest, drop the rest. Fine for points, holes in the map for polygons
- Dissolve (then simplify) : merge what's inside each child cell, then simplify the result. Keeps count and area. Made for polygons.

simplify

same features, fewer vertices

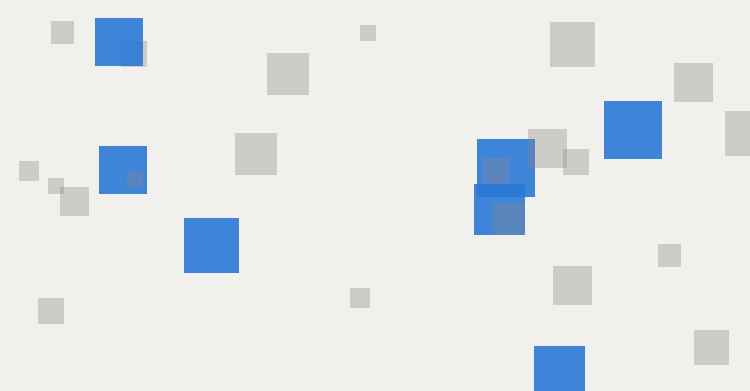


Douglas-Peucker, tolerance = $gsd / 2$
auto picks it for: lines; fixes complexity

cadastral test: 224% overhead, 390 MB state view

prune

fewer features, largest first

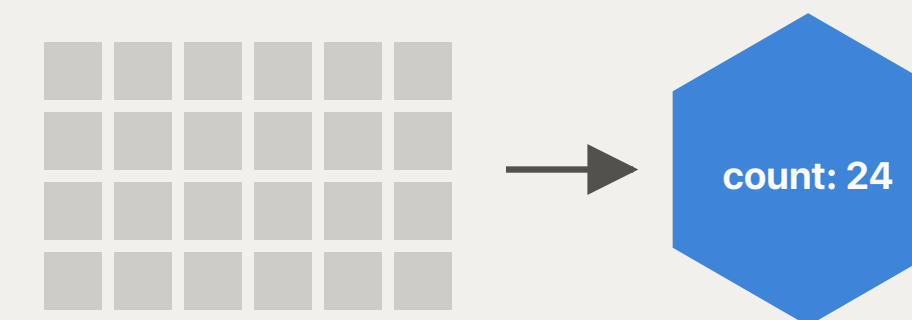


keep 1/4 per level, largest first
auto picks it for: points; fixes count

cadastral test: 58% overhead, 78 MB state view

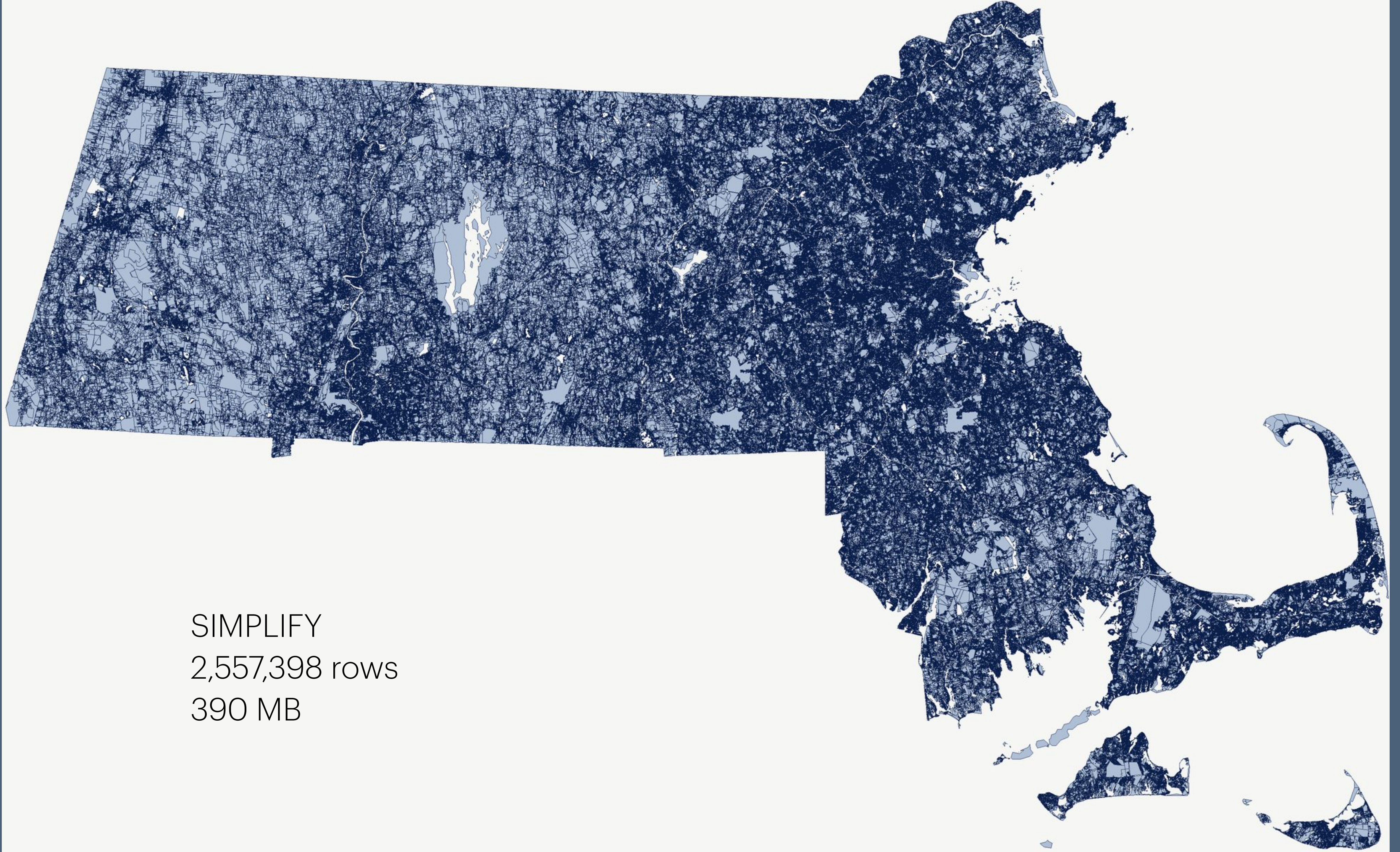
dissolve

union by child cell, summary attributes



count, area_sum, then one simplify pass
auto picks it for: polygons; fixes both

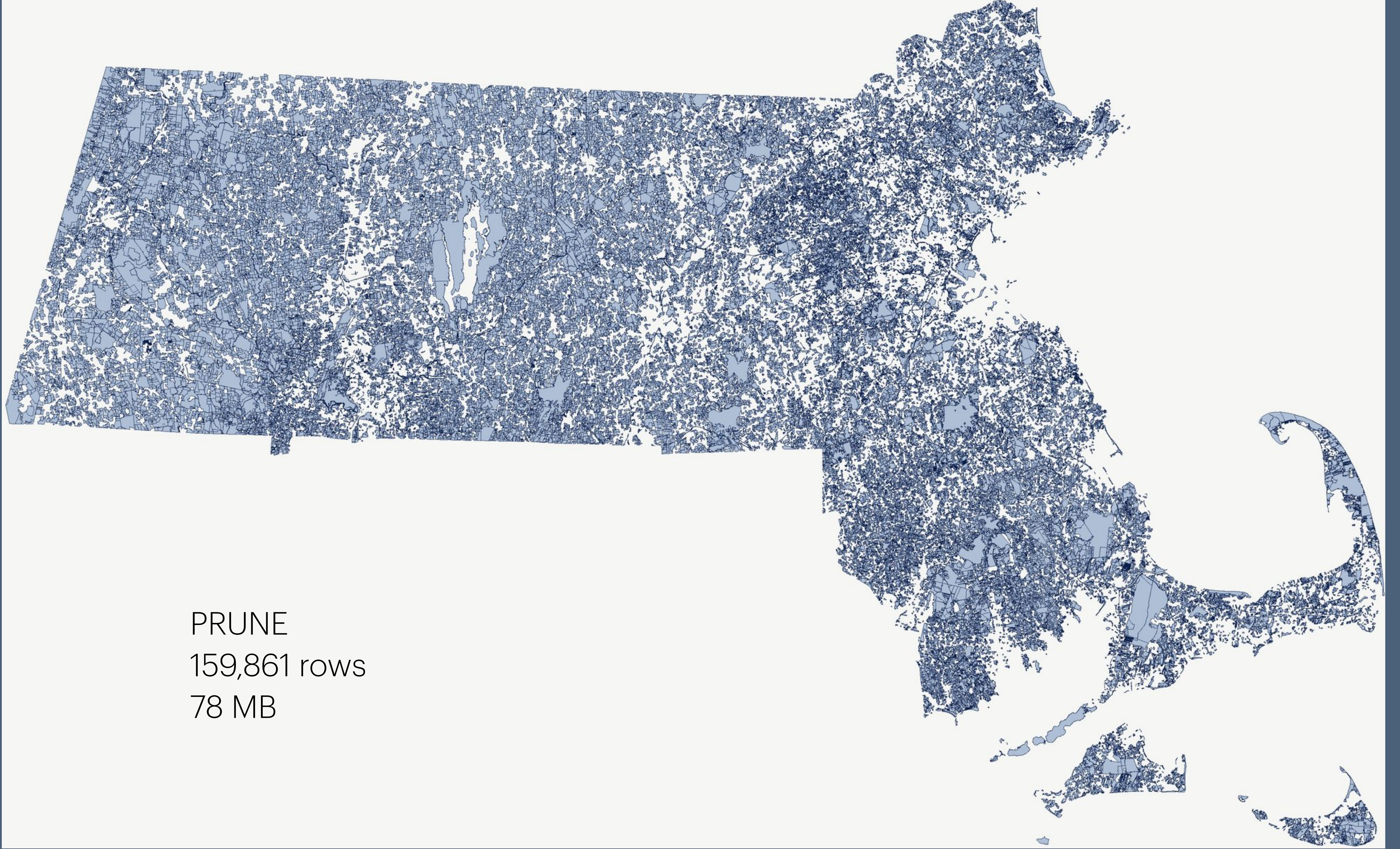
cadastral test: 6% overhead, 10 MB state view



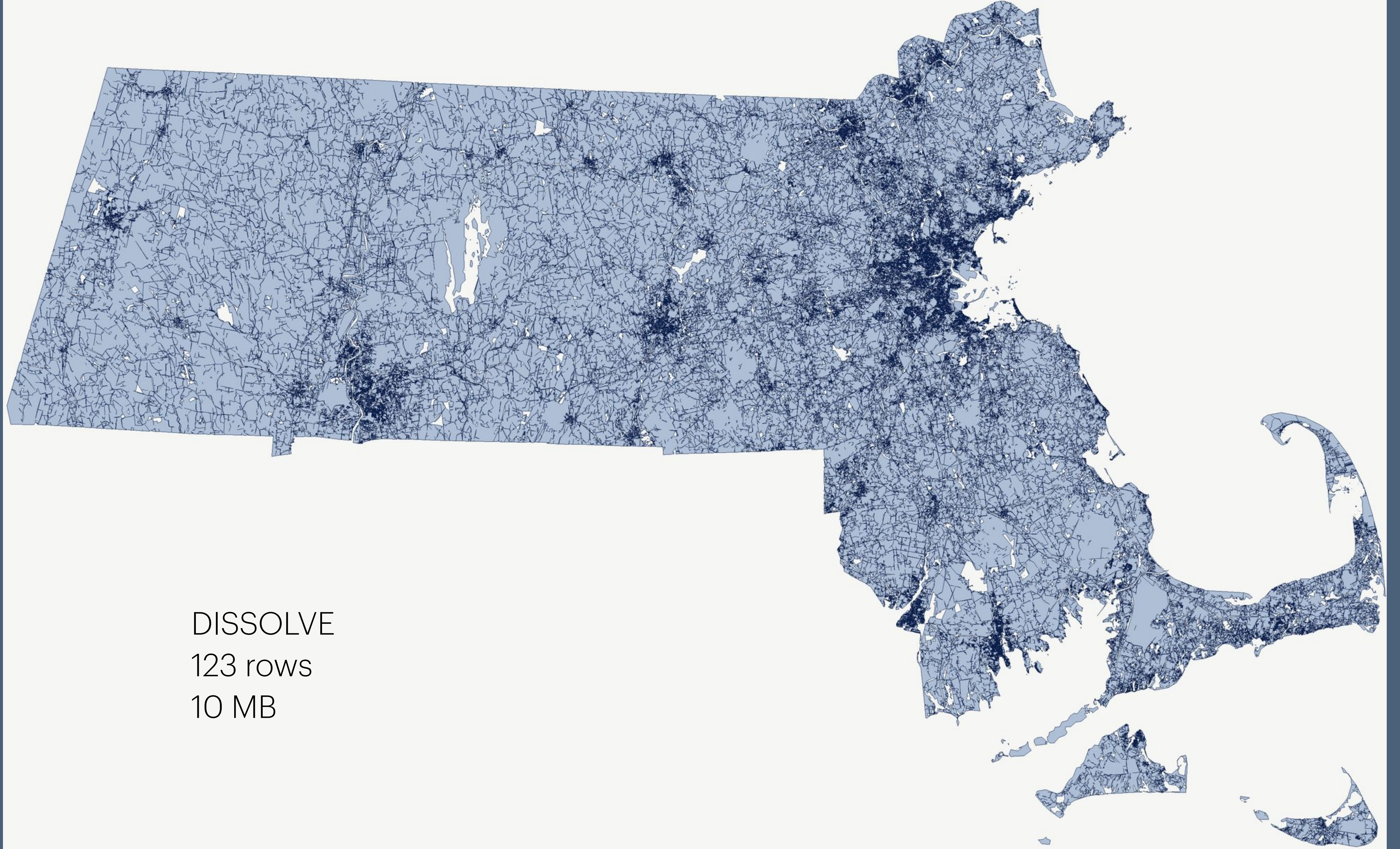
SIMPLIFY

2,557,398 rows

390 MB



PRUNE
159,861 rows
78 MB



DISSOLVE
123 rows
10 MB

What H3 really gave us

Hexagons as plumbing...

- It is a file name
- It knows its parent and its seven children : that's the pyramid
- It is a GROUP BY key, at every scale

"What we did not have to build : no index file, no tile scheme, no lookup service"

And the person looking at the map never sees a hexagon !

What this is not

Let's stay humble...

- Not a replacement for vector tiles (like PMTiles). For a web map they are great, use them
- Not a standard. One convention, one dataset, one reader (mine)
- Not finished

Questions I don't have the answer to

Maybe you do...

- Should overviews live in the GeoParquet spec, or stay a convention or a specific implementation like this one on top of it ?
- Hexagons, or any grid with parents and children ? I like hexagons for their regular pattern, six neighbors, all at the same distance. And honestly, irregularities don't really matter here.
- How to process many columns with different aggregation rules ? That can be painstaking.
- Who writes the second reader ?

We wanted to see the map !

I hope you did...

- **Thank you !**
- GeoPQ Workbench, open source : <https://github.com/gsueur/geopq-workbench>
- Guillaume Sueur, Geomermaids : gsueur@geomermaids.com

