Introduction to Geographic Information Systems (GIS)

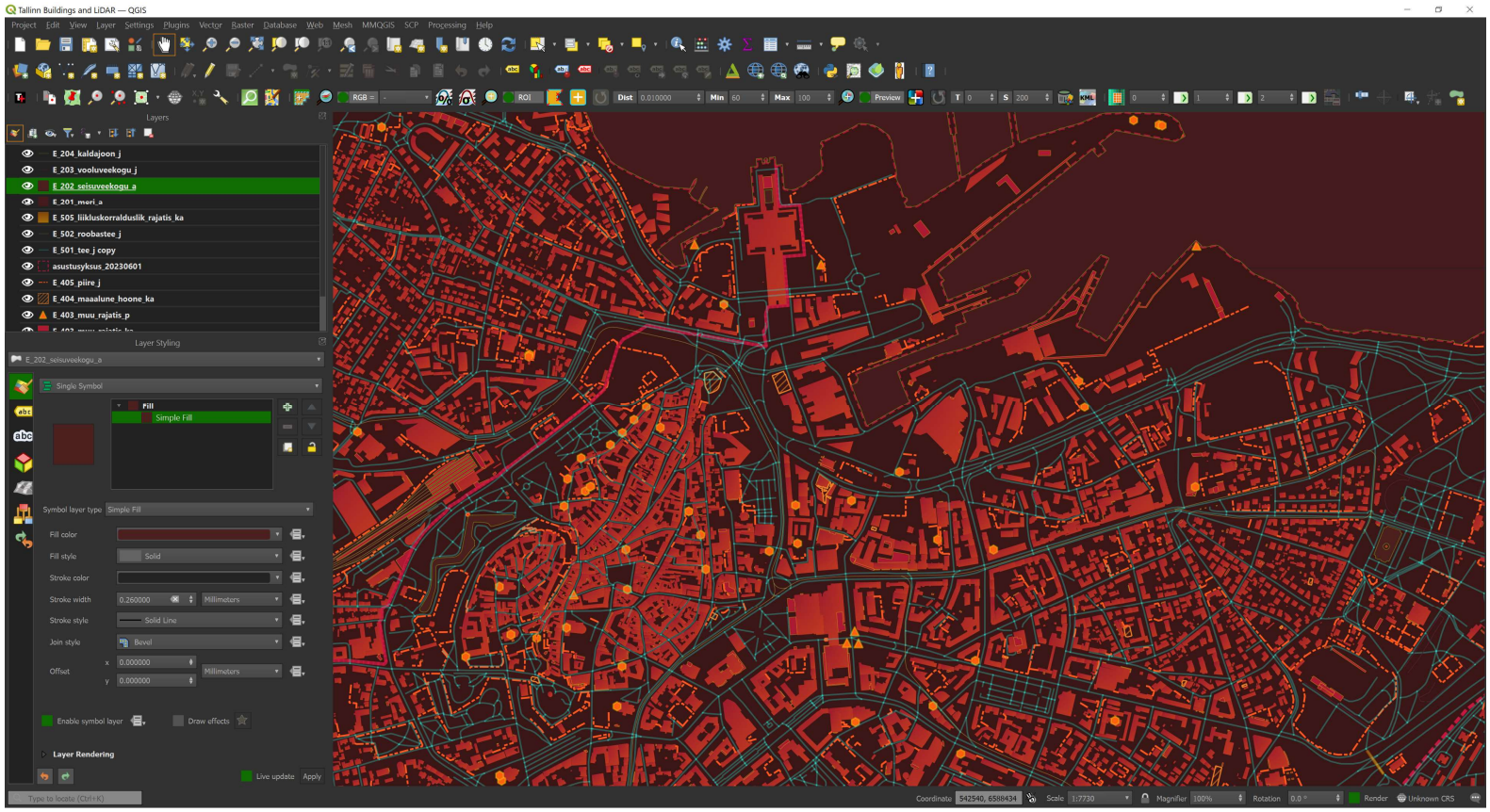
Mark Litwintschik

Green Idea OÜ

Mark Litwintschik

- Author of tech.marksblogg.com
- Consulting for 15+ years
- Big Data, Networking and GIS
- Dual Canadian/British Citizen based in Estonia
- Worked with Banks, Airlines, Telcos, Major Retail Chains, Google, Williams Formula 1

Photoshop for Maps: QGIS

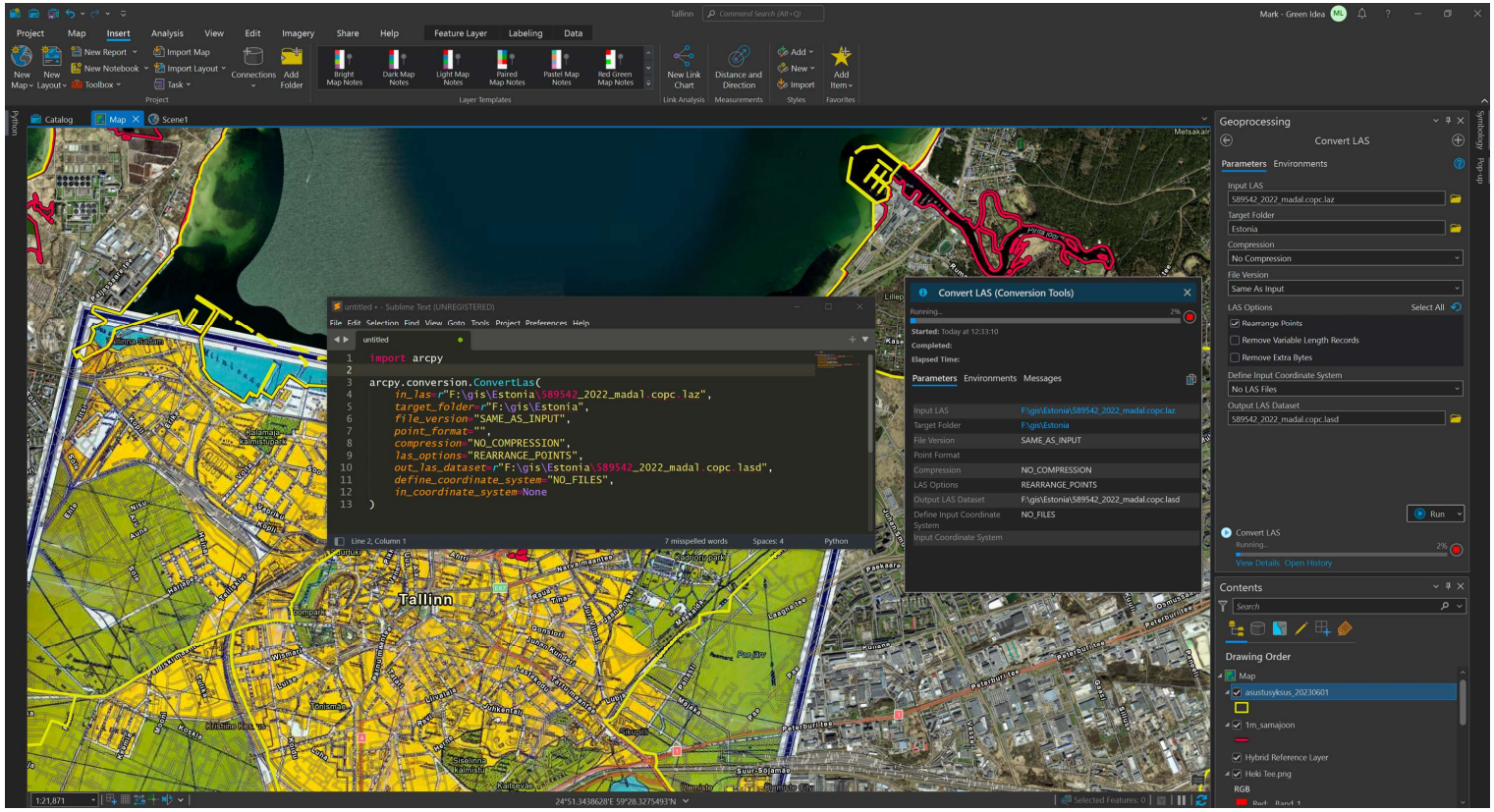


- Free, Open Source
- Drag-and-drop GIS files onto map
- Layers act like Photoshop's
- Python API, venv & REPL
- No GPU acceleration
- Certain tasks have a steep learning curve
- Complex scenes run slower than in ArcGIS Pro **
- 4,371 open tickets on GitHub; 25,421 closed
- Good pairing with Blender for 3D workloads.

** ArcGIS Pro smooth zooming vs QGIS

Credit: The vector data in the above screen shot is from the Estonian Land Board.

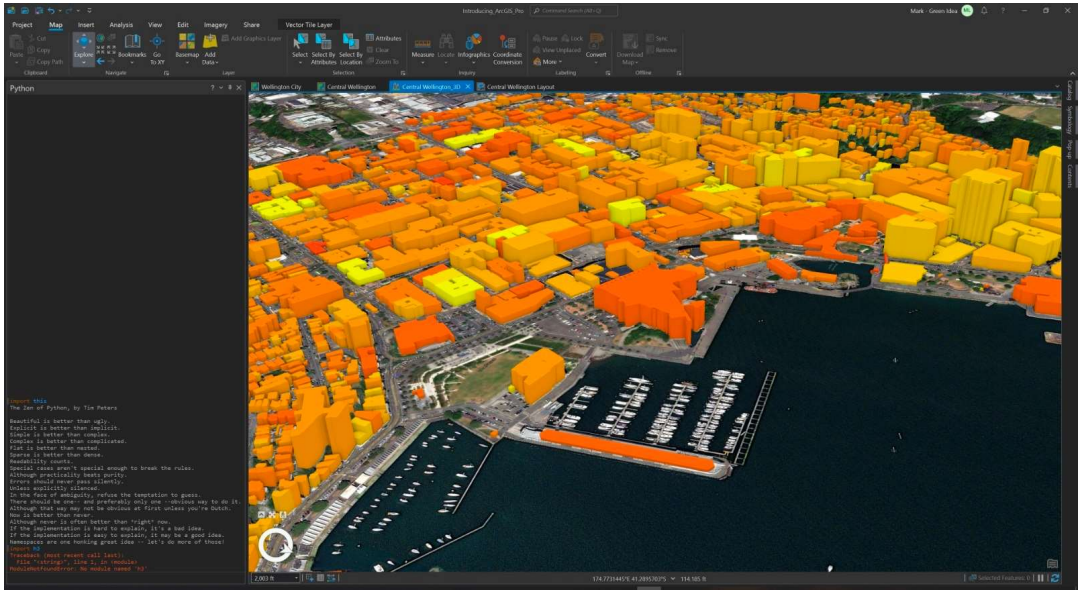
Photoshop for Maps: Esri's ArcGIS Pro



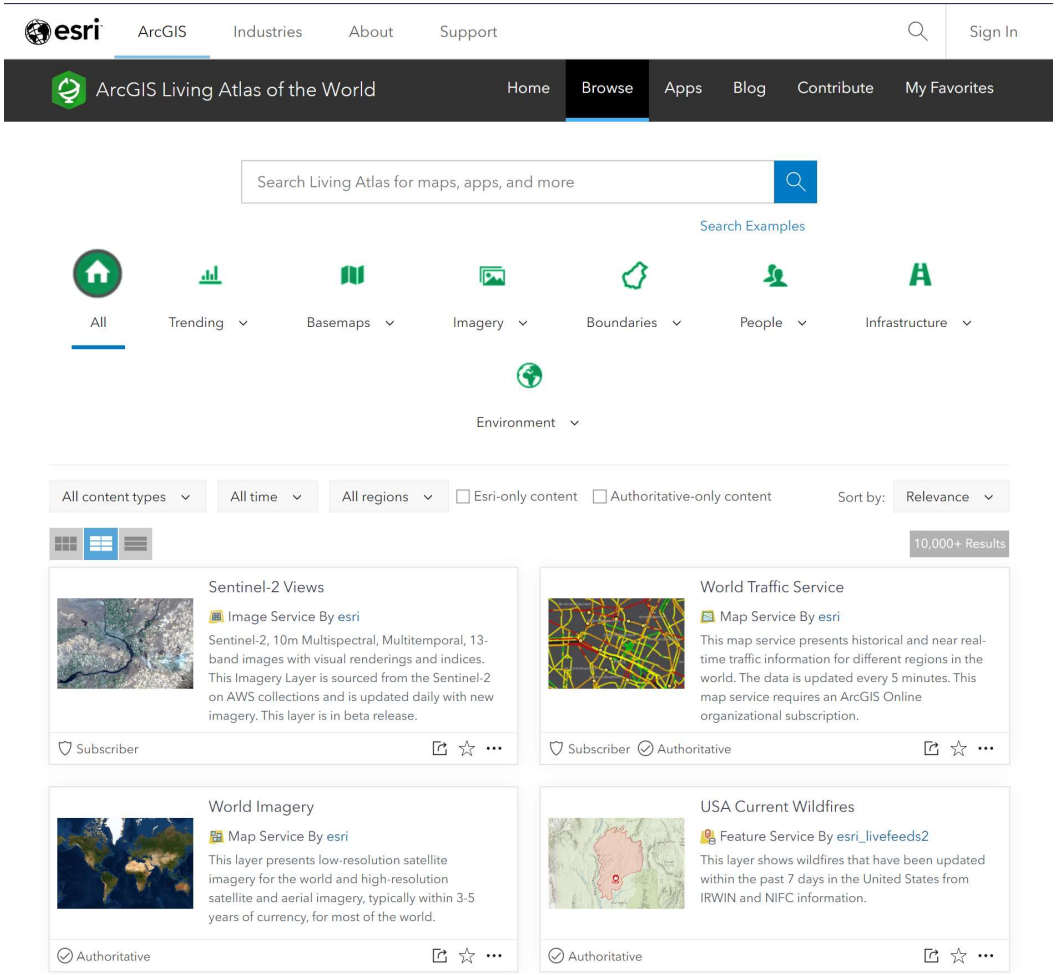
https://www.esri.com/en-us/arcgis/products/arcgis-pro/overview
https://www.youtube.com/playlist?list=PLGZUzt4E4O2IJFxX_Bhp98MJEw5ItRtvb

- $100 / year for students; $800 - $4K / year for pro
- Windows on Desktop; Online for macOS & Linux
- GPU & AVX512 support
- Ribbon Menu
- Many tutorials on YouTube
- Python API & REPL. Create Python code w/ UI
- 734 DLLs vs QGIS' 135
- 5.2 GB footprint is 2x QGIS'
- Uses GDAL 3.4 vs 3.7 for QGIS on Windows
- Most public sector workers don't use the terminal to do day to day GIS work
- 18-20K attendees to their yearly conference in LA

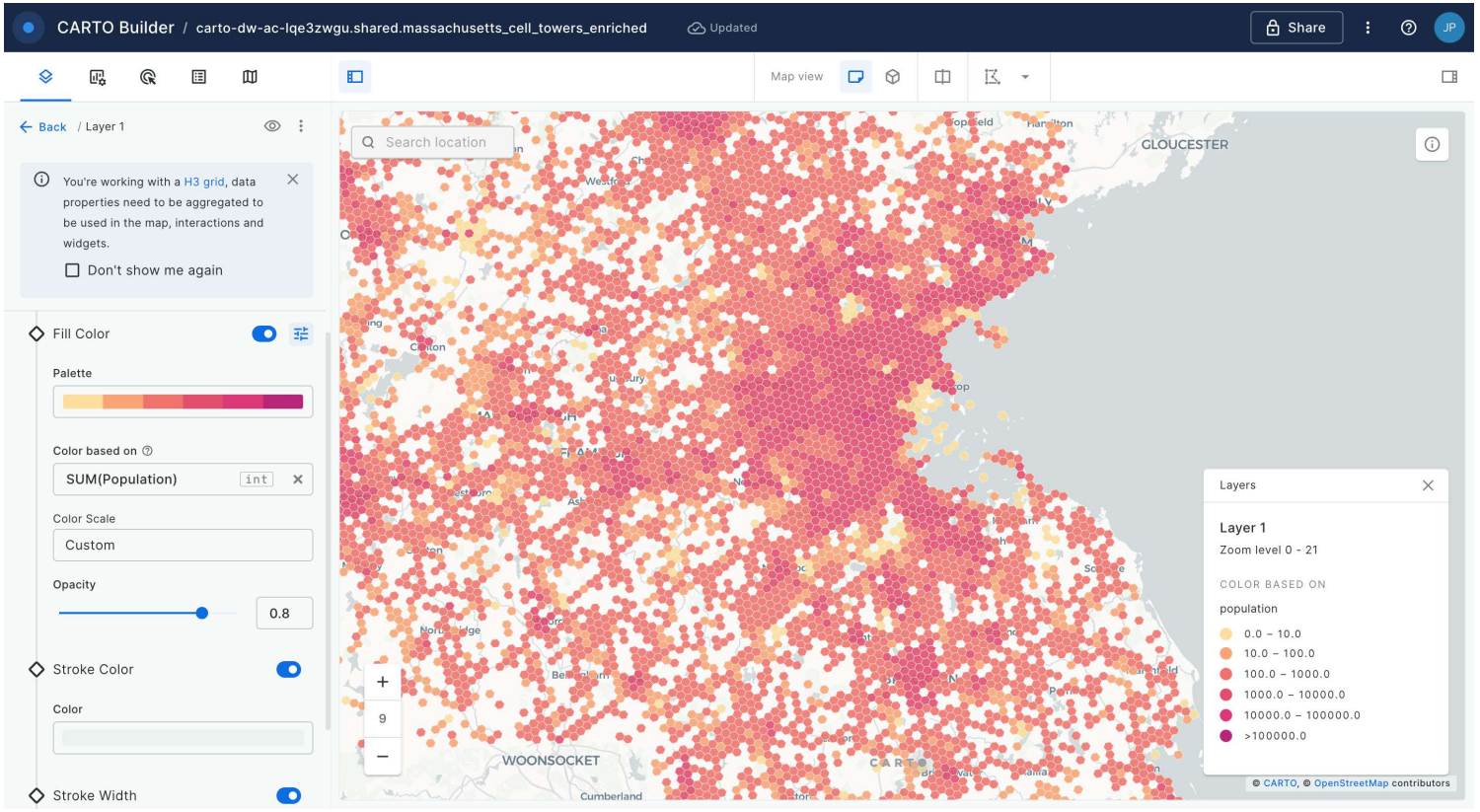
Esri’s ArcGIS Pro: Excellent 3D Support



ArcGIS' Living Atlas' 10K+ GIS Datasets

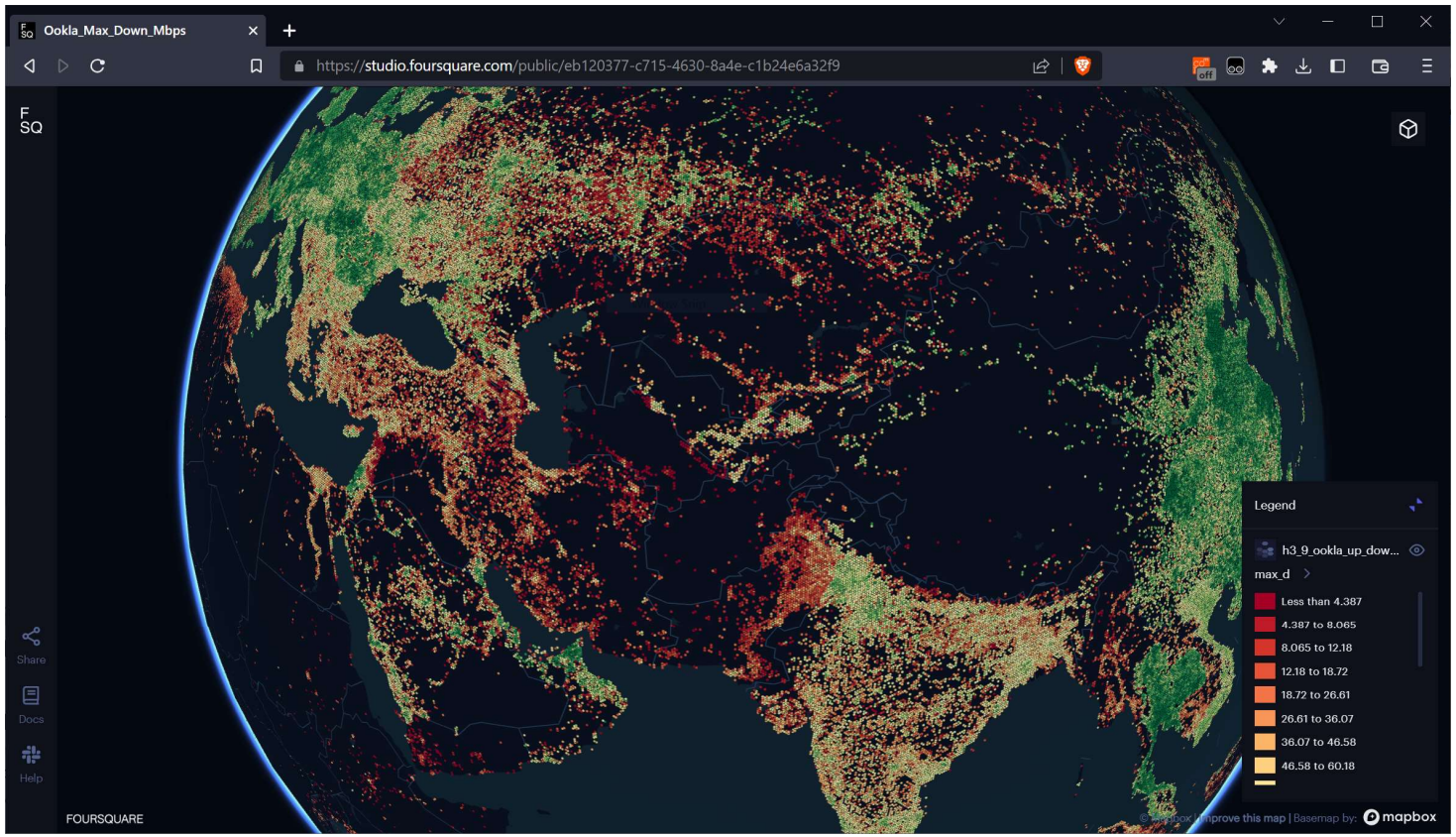


Photoshop for Maps: CARTO



- Started out offering managed PostGIS hosting
- Moved to support BigQuery and DataBricks in past 3-4 years
- Major contributors to https://deck.gl/

Photoshop for Maps: Unfolded

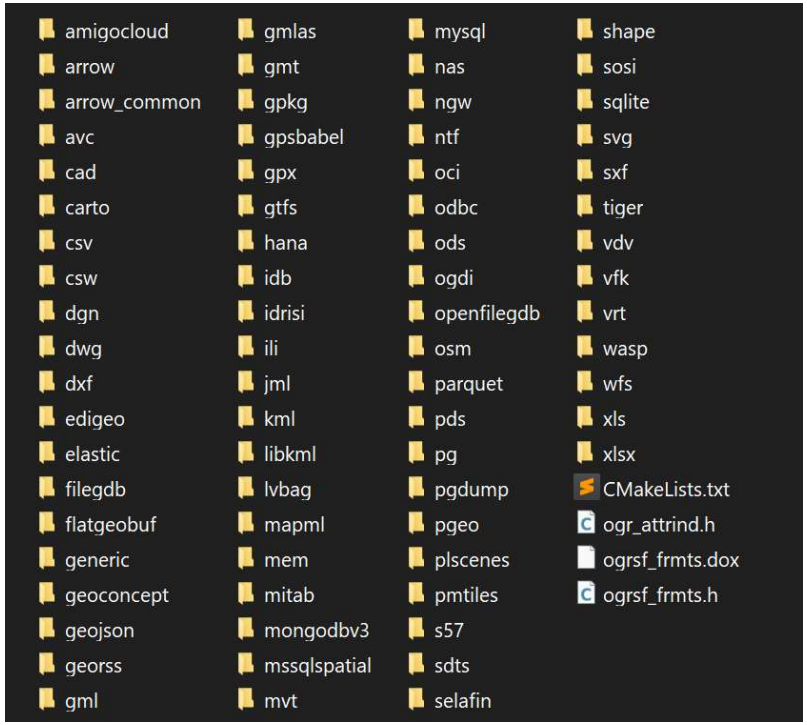


- Many ex-Uber developers; Team behind Uber's H3
- Bought by Foursquare a few years ago
- Major contributors to https://deck.gl/ and almost every H3 database extension and code library

https://studio.foursquare.com/public/eb120377-c715-4630-8a4e-c1b24e6a32f9

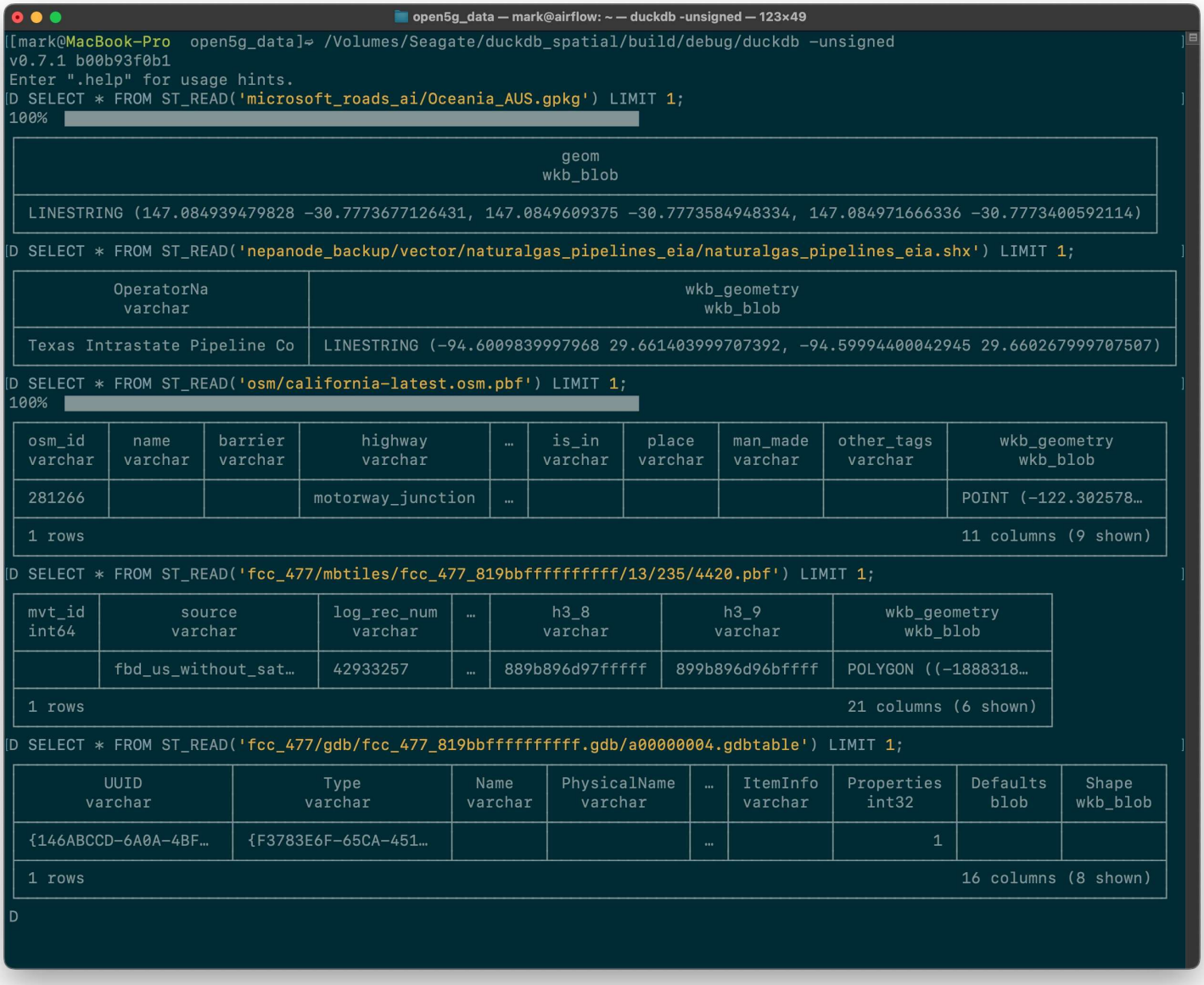
File Formats with GDAL

Provider	Topic	Public or Paid?	curl friendly	Geom Valid?	Months between revisions	Format	Num Rows	Num Cols	CH/PG Size (MB)
Census.gov	2019 TIGER/FACES	Public	Yes	No	Not Refreshed	Shape	20,037,198	50	56,000
ETH Zurich	Global Vegetation: California	Public	Yes	Yes	60	TIFF	20,736,000,000	3	
FCC	Broadband Map Maximums (2022-06	Public	Yes	Yes	6	CSV	877,671,702	23	227,000
FCC	Broadband Map Maximums (2022-12	Public	Yes	Yes	6	CSV	840,553,701	14	
FCC	Broadband Map Minimums (2022-06)	Public	Yes	Yes	6	Shape	115,852,550	16	22,000
FCC	Broadband Map Minimums (2022-12)	Public	Yes	Yes	6	Shape	113,957,747	8	
FCC	Form 477 w/o satellite; 2014 - 2021	Public	No	No	6	CSV	387,126,209	30	606,000
Microsoft	Roads	Public	Yes	No	Not Refreshed	TSV	53,989,010	22	9,800
Microsoft	Rooftops	Public	Yes	Yes	12	GeoJSON	130,099,920	7	89,000
Mozilla	Cell Towers	Public	Yes	Yes	0.1	CSV	144,764,638	25	8,050
Ookla	Cell Performance Public	Public	Yes	Yes	3	Parquet	155,194,705	12	147,000
OpenCellID	Cell Performance	Public	Yes		0.1	CSV	46,509,964	17	23,000
OpenStreetM	Graph (for the entire USA)	Public	Yes	Yes	1	OSM PBF	680,916,190	5	73,000
OpenStreetM	Streets (for the entire USA)	Public	Yes	Yes	1	OSM PBF	13,256,633	8	29,000
Overture	Lines	Public	Yes	Yes	Not Refreshed	OSM PBF	269,900,484	69	92,000
Overture	Points	Public	Yes	Yes	Not Refreshed	PBF	176,109,196	69	14,000
Overture	Polygons	Public	Yes	Yes	Not Refreshed	PBF	819,202,855	69	205,000
Overture	Roads	Public	Yes	Yes	Not Refreshed	OSM PBF	18,356,475	69	12,000
WorldPop	Census: USA	Public	Yes	Yes	24	CSV	15,849,306	6	4,627



https://gdal.org/
http://switchfromshapefile.org/

GDAL & DuckDB


open5g_data — mark@airflow: ~ — duckdb -unsigned — 123×49
[mark@MacBook-Pro open5g_data]↩ /Volumes/Seagate/duckdb_spatial/build/debug/duckdb -unsigned
v0.7.1 b00b93f0b1
Enter ".help" for usage hints.
D SELECT * FROM ST_READ('microsoft_roads_ai/Oceania_AUS.gpkg') LIMIT 1;
100%
geom
wkb_blob
LINESTRING (147.084939479828 -30.7773677126431, 147.0849609375 -30.7773584948334, 147.084971666336 -30.7773400592114)
D SELECT * FROM ST_READ('nepanode_backup/vector/naturalgas_pipelines_eia/naturalgas_pipelines_eia.shx') LIMIT 1;
OperatorNa
varchar
wkb_geometry
wkb_blob
Texas Intrastate Pipeline Co
LINESTRING (-94.6009839997968 29.661403999707392, -94.59994400042945 29.660267999707507)
D SELECT * FROM ST_READ('osm/california-latest.osm.pbf') LIMIT 1;
100%
osm_id varchar
name varchar
barrier varchar
highway varchar
…
is_in varchar
place varchar
man_made varchar
other_tags varchar
wkb_geometry wkb_blob
281266
motorway_junction
…
POINT (-122.302578…
1 rows
11 columns (9 shown)
D SELECT * FROM ST_READ('fcc_477/mbtiles/fcc_477_819bbfffffffff/13/235/4420.pbf') LIMIT 1;
mvt_id int64
source varchar
log_rec_num varchar
…
h3_8 varchar
h3_9 varchar
wkb_geometry wkb_blob
fbd_us_without_sat…
42933257
…
889b896d97fffff
899b896d96bffff
POLYGON ((-1888318…
1 rows
21 columns (6 shown)
D SELECT * FROM ST_READ('fcc_477/gdb/fcc_477_819bbfffffffff.gdb/a00000004.gdbtable') LIMIT 1;
UUID varchar
Type varchar
Name varchar
PhysicalName varchar
…
ItemInfo varchar
Properties int32
Defaults blob
Shape wkb_blob
{146ABCCD-6A0A-4BF…
{F3783E6F-65CA-451…
…
1
1 rows
16 columns (8 shown)
D

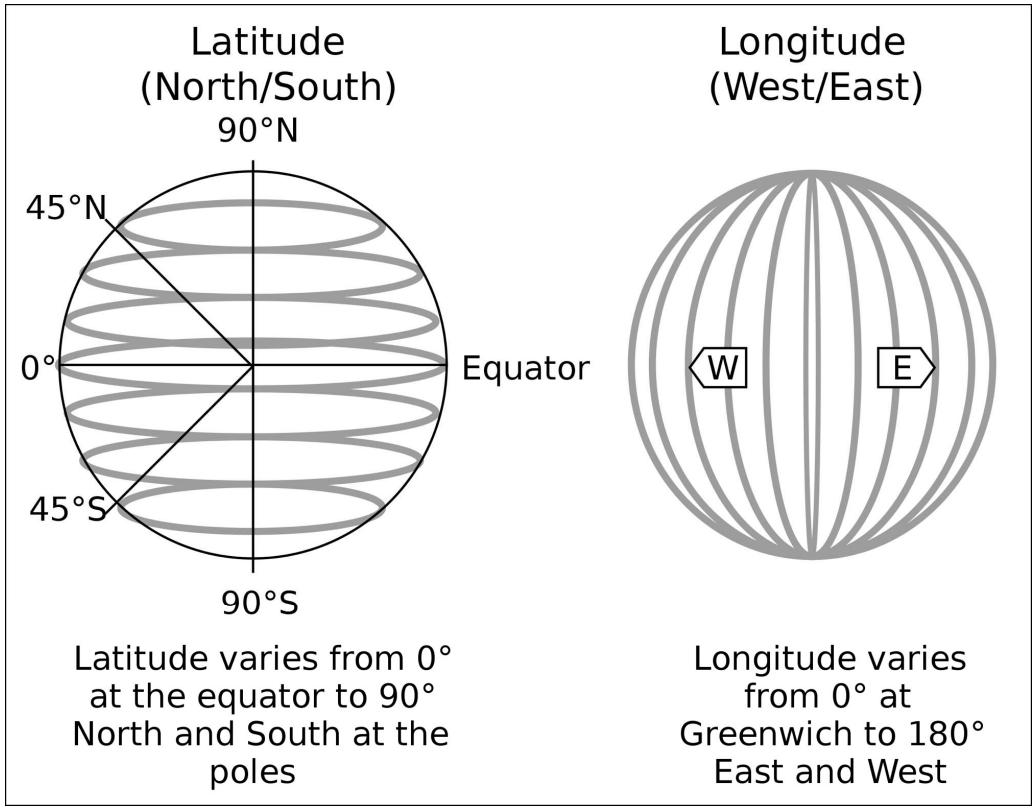
GDAL & DuckDB

```
D select * from st_drivers() order by 2;
```

short_name varchar	long_name varchar	can_create boolean	can_copy boolean	can_open boolean	help_url varchar
AmigoCloud	AmigoCloud	true	false	true	https://gdal.org/drivers/vector/amigocloud.…
AVCBin	Arc/Info Binary Co…	false	false	true	https://gdal.org/drivers/vector/avcbin.html
AVCE00	Arc/Info E00 (ASCI…	false	false	true	https://gdal.org/drivers/vector/avce00.html
DXF	AutoCAD DXF	true	false	true	https://gdal.org/drivers/vector/dxf.html
CAD	AutoCAD Driver	false	false	true	https://gdal.org/drivers/vector/cad.html
Carto	Carto	true	false	true	https://gdal.org/drivers/vector/carto.html
CSV	Comma Separated Va…	true	false	true	https://gdal.org/drivers/vector/csv.html
VFK	Czech Cadastral Ex…	false	false	true	https://gdal.org/drivers/vector/vfk.html
OpenFileGDB	ESRI FileGDB	true	false	true	https://gdal.org/drivers/vector/openfilegdb…
ESRI Shapefile	ESRI Shapefile	true	false	true	https://gdal.org/drivers/vector/shapefile.h…
ESRIJSON	ESRIJSON	false	false	true	https://gdal.org/drivers/vector/esrijson.html
Elasticsearch	Elastic Search	true	false	true	https://gdal.org/drivers/vector/elasticsear…
FlatGeobuf	FlatGeobuf	true	false	true	https://gdal.org/drivers/vector/flatgeobuf.…
EDIGEO	French EDIGEO exch…	false	false	true	https://gdal.org/drivers/vector/edigeo.html
OGR_GMT	GMT ASCII Vectors …	true	false	true	https://gdal.org/drivers/vector/gmt.html
GPSBabel	GPSBabel	true	false	true	https://gdal.org/drivers/vector/gpsbabel.html
GPX	GPX	true	false	true	https://gdal.org/drivers/vector/gpx.html
GeoJSON	GeoJSON	true	false	true	https://gdal.org/drivers/vector/geojson.html
GeoJSONSeq	GeoJSON Sequence	true	false	true	https://gdal.org/drivers/vector/geojsonseq.…
GPKG	GeoPackage	true	true	true	https://gdal.org/drivers/vector/gpkg.html
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
DGN	Microstation DGN	true	false	true	https://gdal.org/drivers/vector/dgn.html
NGW	NextGIS Web	true	true	true	https://gdal.org/drivers/vector/ngw.html
OAPIF	OGC API - Features	false	false	true	https://gdal.org/drivers/vector/oapif.html
CSW	OGC CSW (Catalog …	false	false	true	https://gdal.org/drivers/vector/csw.html
WFS	OGC WFS (Web Featu…	false	false	true	https://gdal.org/drivers/vector/wfs.html
ODS	Open Document/ Lib…	true	false	true	https://gdal.org/drivers/vector/ods.html
JML	OpenJUMP JML	true	false	true	https://gdal.org/drivers/vector/jml.html
OSM	OpenStreetMap XML …	false	false	true	https://gdal.org/drivers/vector/osm.html
PLSCENES	Planet Labs Scenes…	false	false	true	https://gdal.org/drivers/vector/plscenes.html
PGDUMP	PostgreSQL SQL dump	true	false	false	https://gdal.org/drivers/vector/pgdump.html
SQLite	SQLite / Spatialite	true	false	true	https://gdal.org/drivers/vector/sqlite.html
SVG	Scalable Vector Gr…	false	false	true	https://gdal.org/drivers/vector/svg.html
Selafin	Selafin	true	false	true	https://gdal.org/drivers/vector/selafin.html
SXF	Storage and eXchan…	false	false	true	https://gdal.org/drivers/vector/sxf.html
TopoJSON	TopoJSON	false	false	true	https://gdal.org/drivers/vector/topojson.html
TIGER	U.S. Census TIGER/…	false	false	true	https://gdal.org/drivers/vector/tiger.html
UK .NTF	UK .NTF	false	false	true	https://gdal.org/drivers/vector/ntf.html
VDV	VDV-451/VDV-452/IN…	true	false	true	https://gdal.org/drivers/vector/vdv.html
OGR_VRT	VRT - Virtual Data…	false	false	true	https://gdal.org/drivers/vector/vrt.html
WAsP	WAsP .map format	true	false	true	https://gdal.org/drivers/vector/wasp.html

51 rows (40 shown) 6 columns

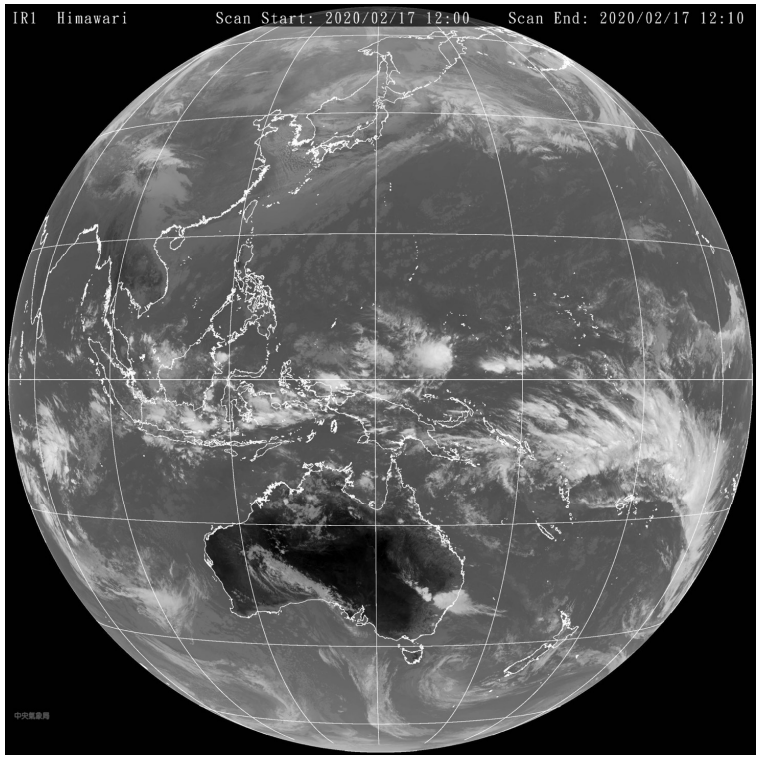
Basics: Longitude, Latitude, Altitude



Location	Latitude	Longitude
Tallinn, Estonia	59.5° N	24.8° E
San Francisco	37.8° N	122.4° W
Null Island	0°N	0°E

https://en.wikipedia.org/wiki/Geographic_coordinate_system

Basics: Raster, Vector and Point Clouds

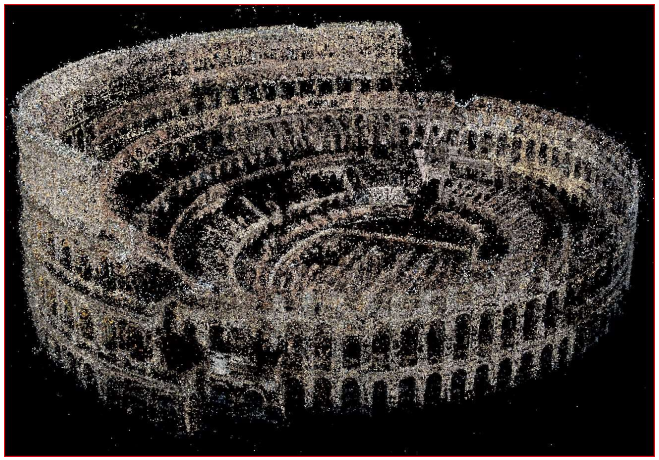


Geometry primitives (2D)

Type	Examples	
Point		`POINT (30 10)`
LineString		`LINESTRING (30 10, 10 30, 40 40)`
Polygon		`POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))`
		`POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 35, 30 20, 20 30))`

Multipart geometries (2D)

Type	Examples	
MultiPoint		`MULTIPOINT ((10 40), (40 30), (20 20), (30 10))`
		`MULTIPOINT (10 40, 40 30, 20 20, 30 10)`
MultiLineString		`MULTILINESTRING ((10 10, 20 20, 10 40), (40 40, 30 30, 40 20, 30 10))`
MultiPolygon		`MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)), ((15 5, 40 10, 10 20, 5 10, 15 5)))`
		`MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)), ((20 35, 10 30, 10 10, 30 5, 45 20, 20 35), (30 20, 20 15, 20 25, 30 20)))`
GeometryCollection		`GEOMETRYCOLLECTION (POINT (40 10), LINESTRING (10 10, 20 20, 10 40), POLYGON ((40 40, 20 45, 45 30, 40 40)))`



https://carpentries-incubator.github.io/geospatial-python/aio/index.html

Basics: Layers

Labels layer

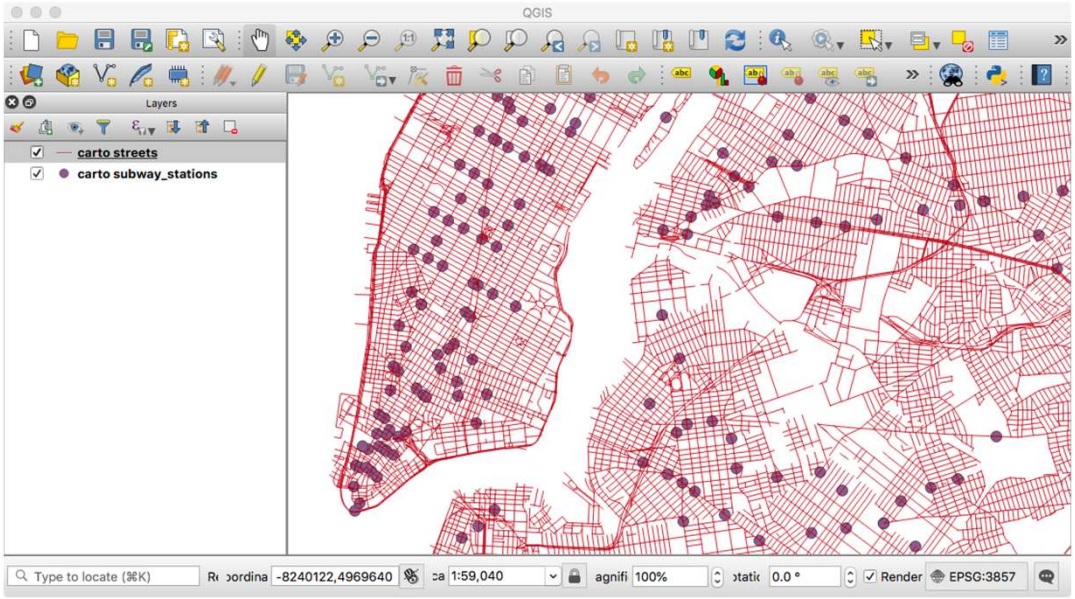
LINESTRINGs road layer

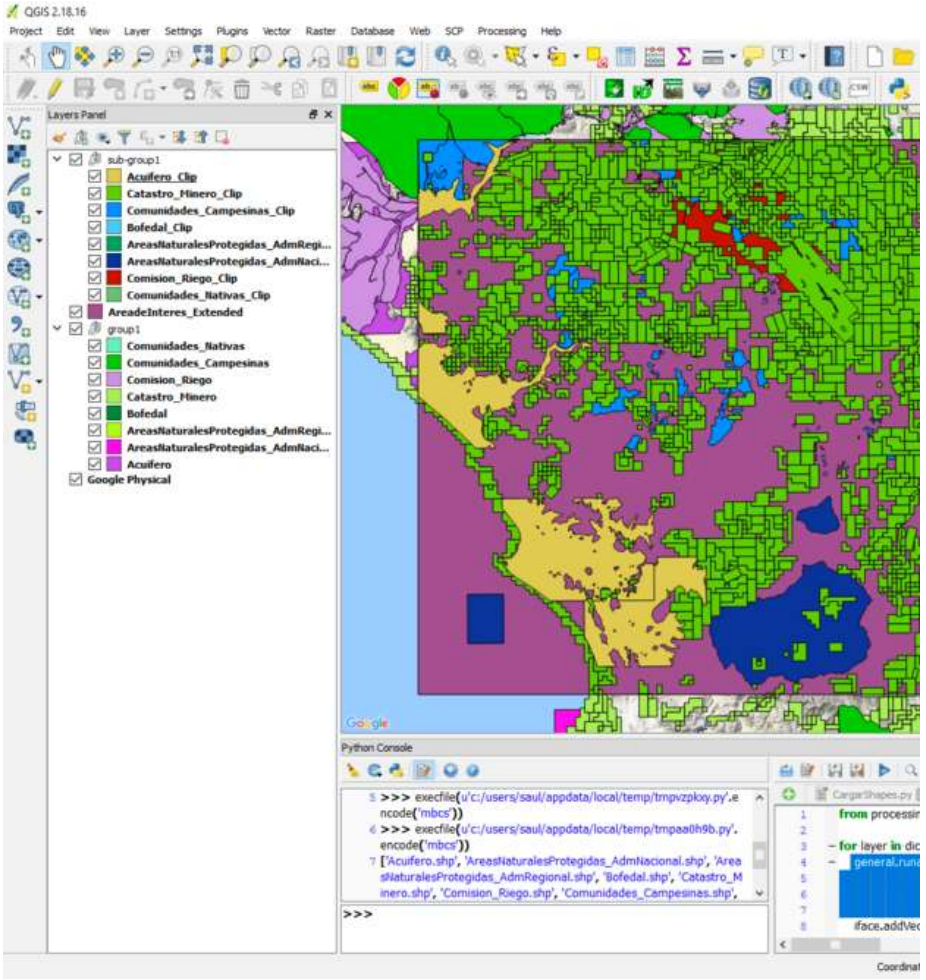
Satellite image basemap

Basics: Layers

Some GIS file formats support layers, others don't and layers are stored on a file basis instead:

versus

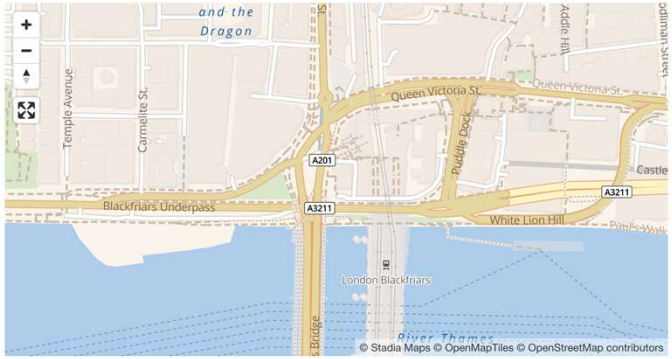




Basics: Map Tiles

Pre-rendered Raster Map Tiles

Vector Map Tiles



Want to customize the styles even more? Need complete control? Our vector map tiles can do it all. With smooth zooming and scrolling, and lighter downloads, our basemap vector tiles allow you to mix and mash your data into the perfect form, with a little flair.

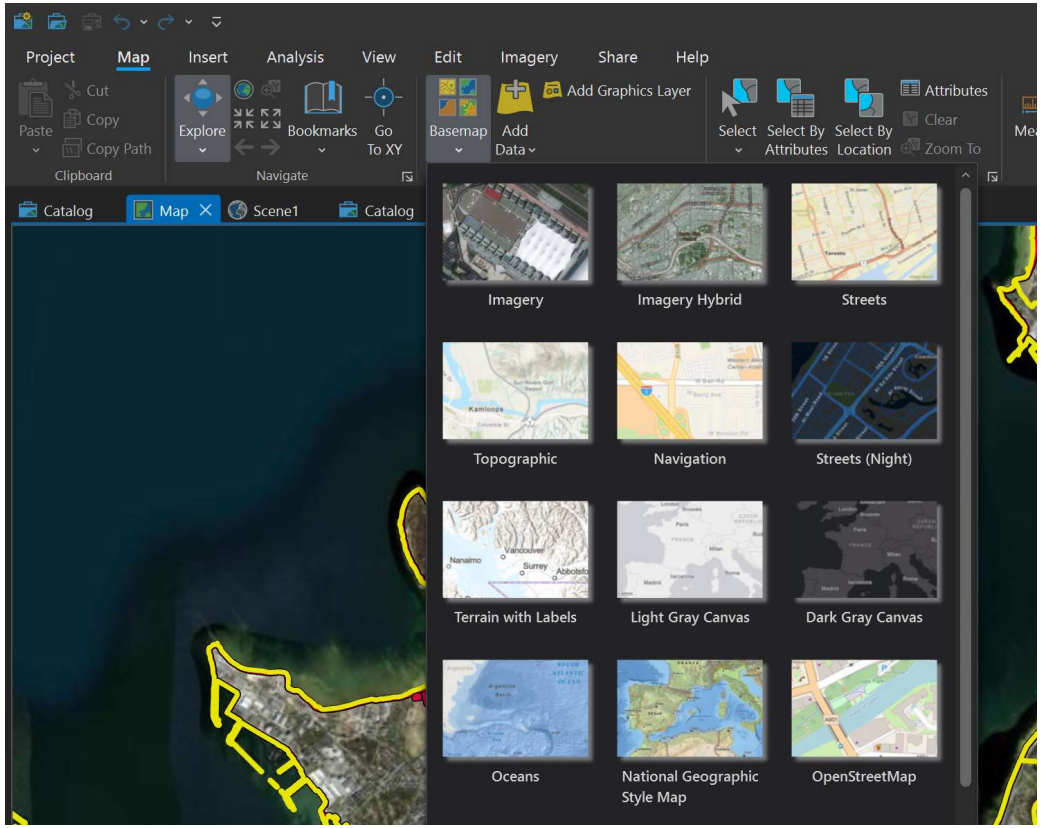
You can use our vector map tiles on the web, desktop, and mobile platforms alike. With excellent support in MapLibre GL JS, OpenLayers, and the Leaflet vector plugin for web, and MapLibre GL Native for Qt and mobile, our vector map tiles allow you to express your map in your style, your way.

Learn more about how to use vector maps in our documentation.

- Smaller
- Flexible Styling

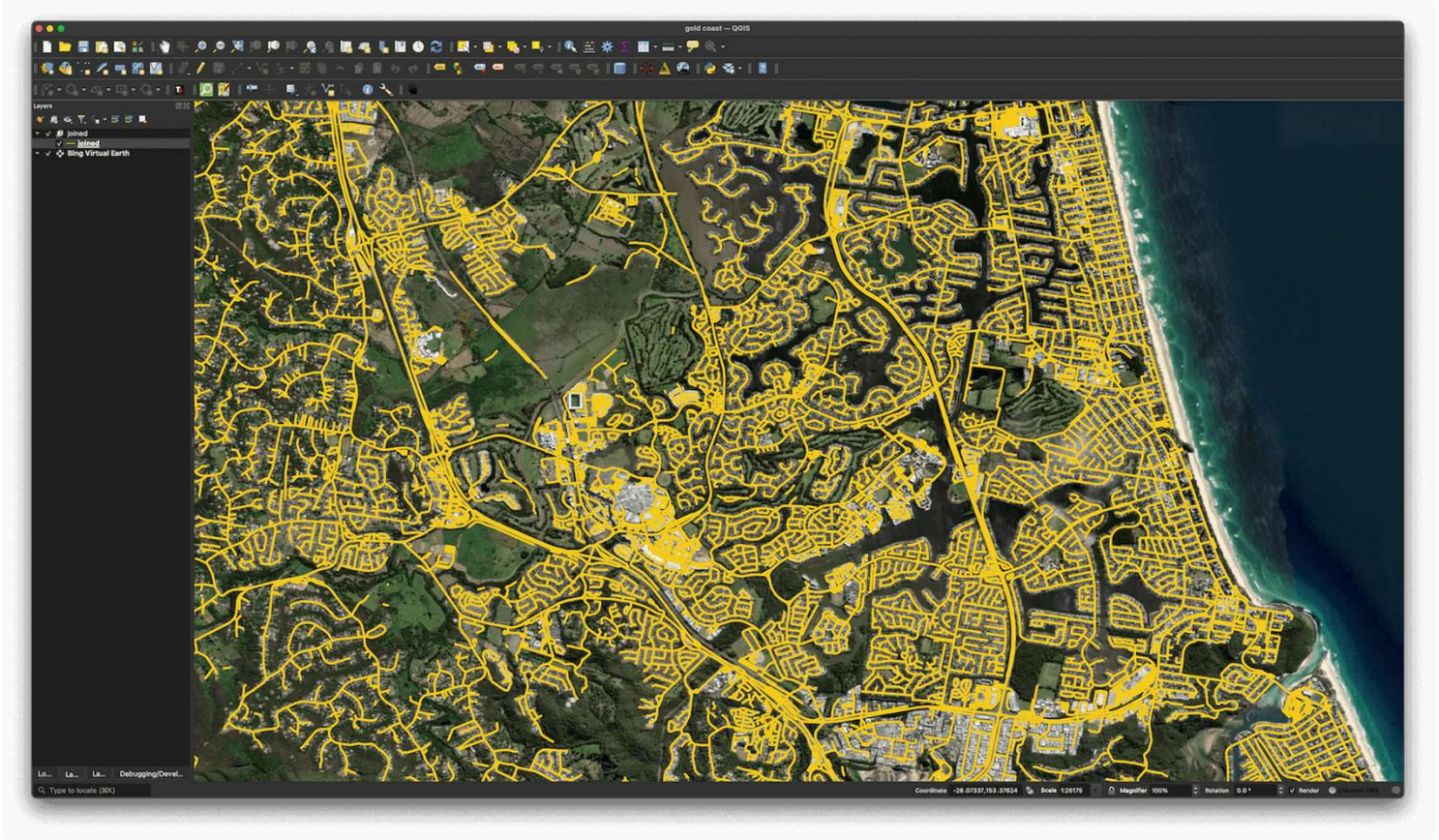
https://stadiamaps.com/products/map-tiles/
https://en.wikipedia.org/wiki/Tiled_web_map
https://en.wikipedia.org/wiki/Vector_tiles

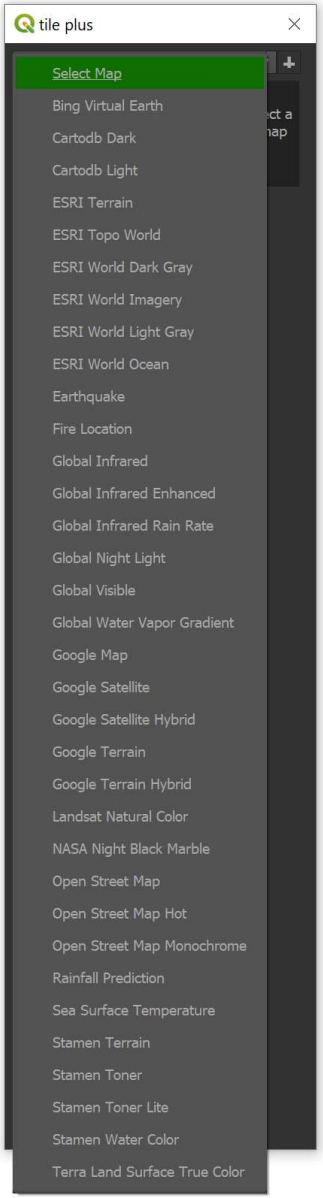
Basics: Basemap Tiles



https://docs.mapbox.com/studio-manual/reference/tilesets/
https://github.com/polygonal-network/tukey/tree/main/docs/mbtiles

Tile+ for QGIS





https://geographicalanalysis.com/download-tile-plus-plugin-qgis/

Tile+ for QGIS

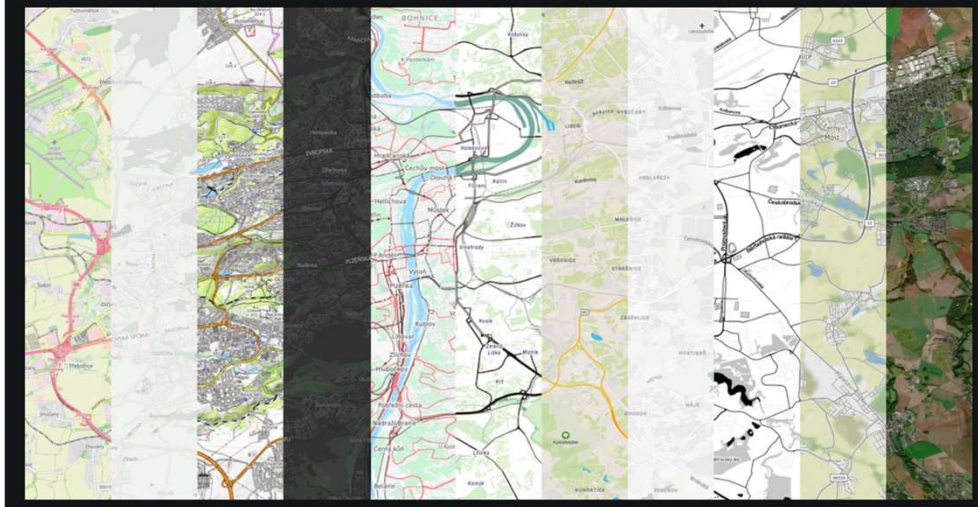
This seems to be well-adopted among tile servers.

```
$ grep -Eo 'https?://.*\]]' /Users/mark/Library/Application\ Support/QGIS/QGIS3/profiles/default/python/plugins/tile_p
    | python3 -c "import sys; from urllib.parse import unquote; print(unquote(sys.stdin.read()));" \
    | sed 's/]]//g' \
    | sort
```

```
http://basemaps.cartocdn.com/dark_all/{z}/{x}/{y}.png
http://basemaps.cartocdn.com/light_all/{z}/{x}/{y}.png
http://ecn.t3.tiles.virtualearth.net/tiles/a{q}.jpeg?g=1
http://realearth.ssec.wisc.edu/tiles/Earthquake-mag/{z}/{x}/{y}.png
http://realearth.ssec.wisc.edu/tiles/GLOBALterratc/{z}/{x}/{y}.png
http://realearth.ssec.wisc.edu/tiles/NESDIS-GHE-HourlyRainfall/{z}/{x}/{y}.png
http://realearth.ssec.wisc.edu/tiles/NESDIS-SST/{z}/{x}/{y}.png
http://realearth.ssec.wisc.edu/tiles/NightLightsColored/{z}/{x}/{y}.png
http://realearth.ssec.wisc.edu/tiles/VIIRS-Fire/{z}/{x}/{y}.png
http://realearth.ssec.wisc.edu/tiles/VIIRS-MASK-54000x27000/{z}/{x}/{y}.png
http://realearth.ssec.wisc.edu/tiles/globalir-avn/{z}/{x}/{y}.png
http://realearth.ssec.wisc.edu/tiles/globalir-rr/{z}/{x}/{y}.png
http://realearth.ssec.wisc.edu/tiles/globalir/{z}/{x}/{y}.png
http://realearth.ssec.wisc.edu/tiles/globalvis/{z}/{x}/{y}.png
http://realearth.ssec.wisc.edu/tiles/globalwv-grad/{z}/{x}/{y}.png
http://realearth.ssec.wisc.edu/tiles/lsat8-llook-fc/{z}/{x}/{y}.png
http://services.arcgisonline.com/ArcGIS/rest/services/Canvas/World_Dark_Gray_Base/MapServer/tile/{z}/{y}/{x}
http://services.arcgisonline.com/ArcGIS/rest/services/Canvas/World_Light_Gray_Base/MapServer/tile/{z}/{y}/{x}
http://services.arcgisonline.com/ArcGIS/rest/services/World_Topo_Map/MapServer/tile/{z}/{y}/{x}
http://tile.openstreetmap.fr/hot/{z}/{x}/{y}.png
http://tile.openstreetmap.org/{z}/{x}/{y}.png
http://tile.stamen.com/terrain/{z}/{x}/{y}.png
http://tile.stamen.com/toner-lite/{z}/{x}/{y}.png
http://tile.stamen.com/toner/{z}/{x}/{y}.png
http://tile.stamen.com/watercolor/{z}/{x}/{y}.jpg
http://tiles.wmflabs.org/bw-mapnik/{z}/{x}/{y}.png
https://mt1.google.com/vt/lyrs=m&x={x}&y={y}&z={z}
https://mt1.google.com/vt/lyrs=p&x={x}&y={y}&z={z}
https://mt1.google.com/vt/lyrs=s&x={x}&y={y}&z={z}
https://mt1.google.com/vt/lyrs=t&x={x}&y={y}&z={z}
https://mt1.google.com/vt/lyrs=y&x={x}&y={y}&z={z}
https://server.arcgisonline.com/ArcGIS/rest/services/World_Imagery/MapServer/tile/{z}/{y}/{x}
https://server.arcgisonline.com/ArcGIS/rest/services/World_Terrain_Base/MapServer/tile/{z}/{y}/{x}
https://services.arcgisonline.com/ArcGIS/rest/services/Ocean/World_Ocean_Base/MapServer/tile/{z}/{y}/{x}
```

xyzservices

Source of XYZ tiles providers.

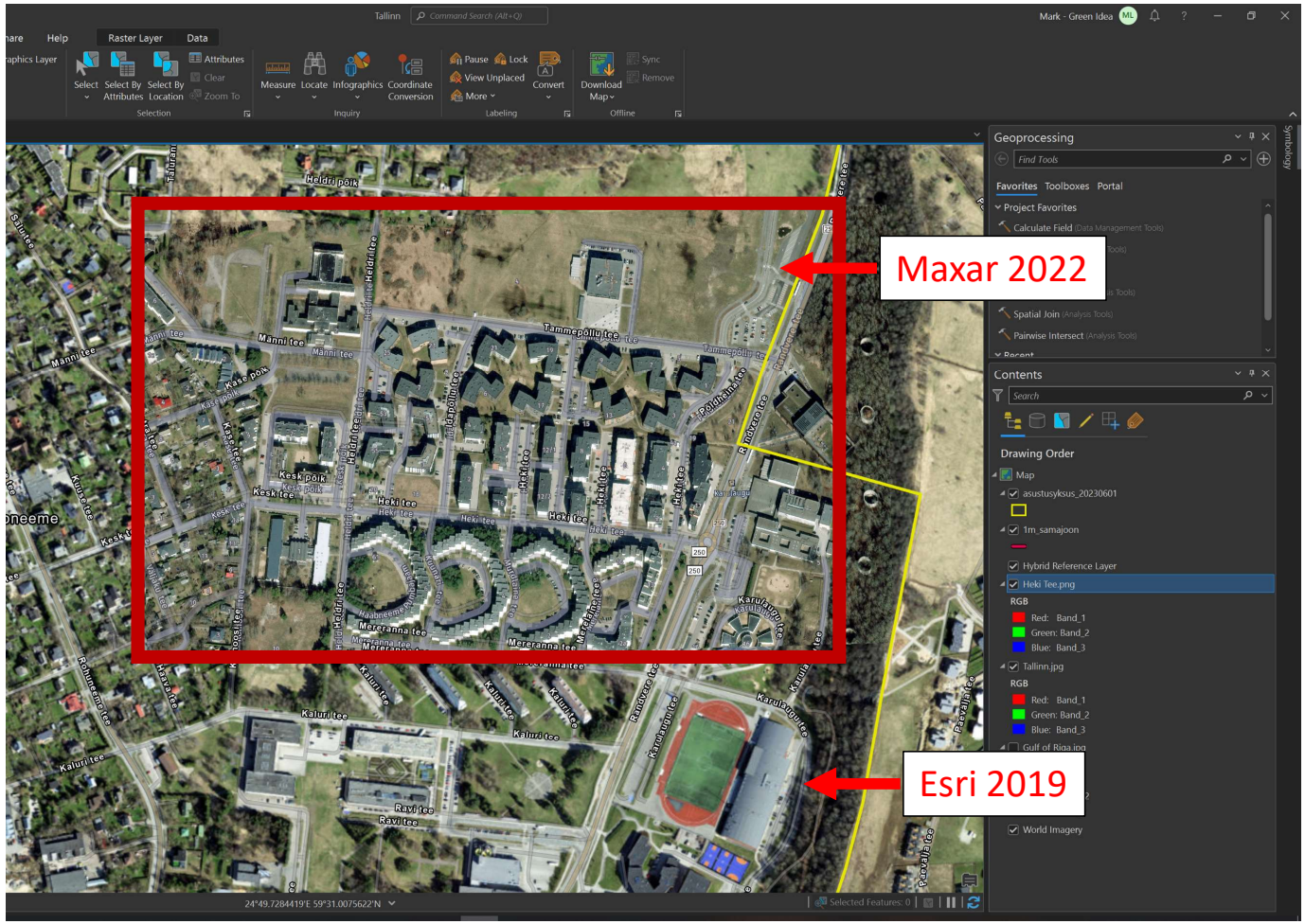


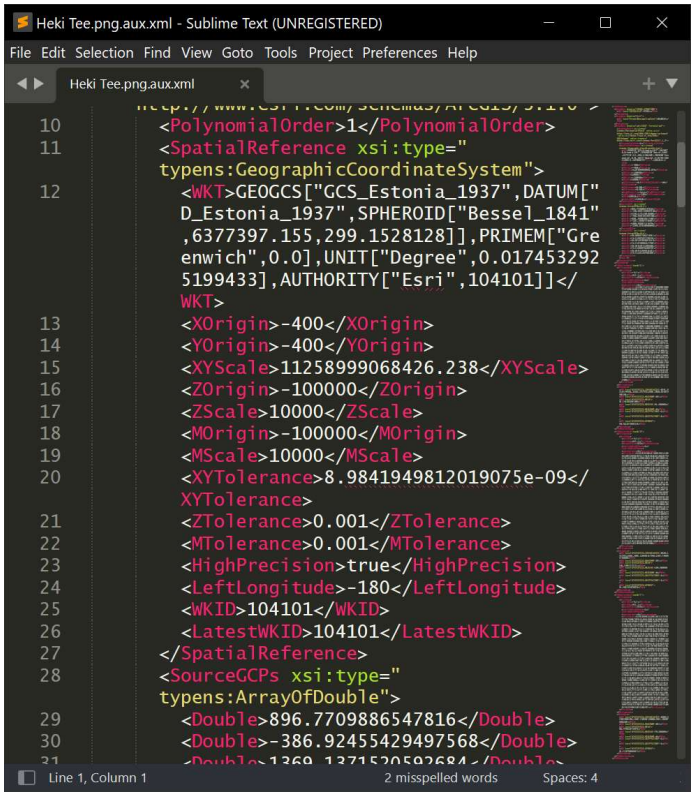
`xyzservices` is a lightweight library providing a repository of available XYZ services offering raster basemap tiles. The repository is provided via Python API and as a compressed JSON file.

XYZ tiles can be used as background for your maps to provide necessary spatial context. `xyzservices` offer specifications of many tile services and provide an easy-to-use tools to plug them into your work, no matter if interactive or static.

https://geographicalanalysis.com/download-tile-plus-plugin-qgis/

Layering Satellite Imagery





Well-Known Binary (WKB), WKT & GeoJSON

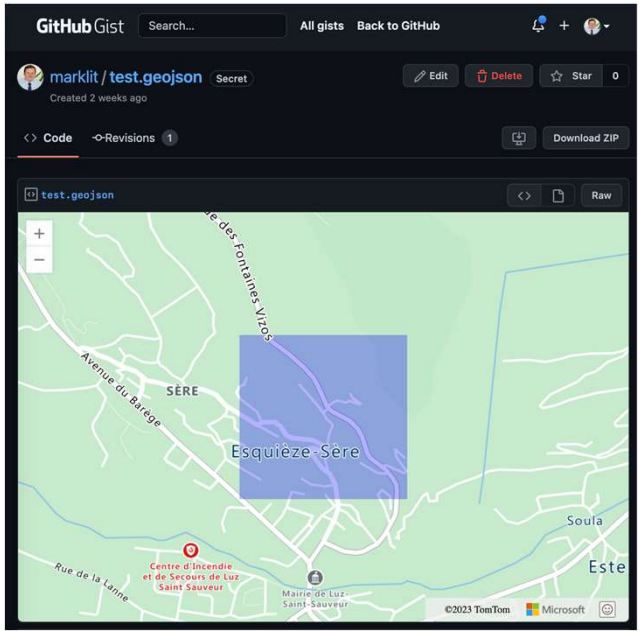
```
WKB:

    0103000000010000000500000000000000008076
    bf35b41b7fa3704540000000000000035b41b
    7fa3704540000000000000000f6197a971f7045
    4000000000008076bff6197a971f704540000000
    00008076bf35b41b7fa3704540

WKT:

    POLYGON((-0.0054931640625 42.8799895177148,
             0 42.8799895177148,
             0 42.8759641023825,
            -0.0054931640625 42.8759641023825,
            -0.0054931640625 42.8799895177148))

GeoJSON:

    {
        "type": "Polygon",
        "coordinates": [
            [
                [-0.0054931640625, 42.8799895177148],
                [0, 42.8799895177148],
                [0, 42.8759641023825],
                [-0.0054931640625, 42.8759641023825],
                [-0.0054931640625, 42.8799895177148]
            ]
        ]
    }
```

Geometry primitives (2D)

Type	Examples
Point	`POINT (30 10)`
LineString	`LINESTRING (30 10, 10 30, 40 40)`
Polygon	`POLYGON ((30 10, 40 40, 20 40, 10 20, 30 10))`
	`POLYGON ((35 10, 45 45, 15 40, 10 20, 35 10), (20 30, 35 35, 30 20, 20 30))`

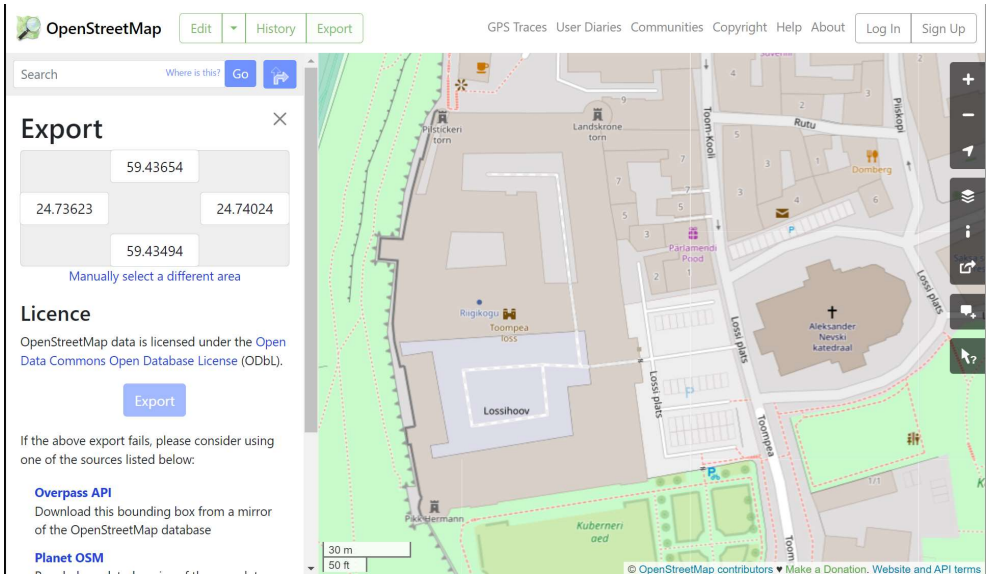
Multipart geometries (2D)

Type	Examples
MultiPoint	`MULTIPOINT ((10 40), (40 30), (20 20), (30 10))`
	`MULTIPOINT (10 40, 40 30, 20 20, 30 10)`
MultiLineString	`MULTILINESTRING ((10 10, 20 20, 10 40), (40 40, 30 30, 40 20, 30 10))`
MultiPolygon	`MULTIPOLYGON (((30 20, 45 40, 10 40, 30 20)), ((15 5, 40 10, 10 20, 5 10, 15 5)))`
	`MULTIPOLYGON (((40 40, 20 45, 45 30, 40 40)), ((20 35, 10 30, 10 10, 30 5, 45 20, 20 35), (30 20, 20 15, 20 25, 30 20)))`
GeometryCollection	`GEOMETRYCOLLECTION (POINT (40 10), LINESTRING (10 10, 20 20, 10 40), POLYGON ((40 40, 20 45, 45 30, 40 40)))`



https://rodic.fr/wp-content/uploads/2015/11/geom_converter.html
https://en.wikipedia.org/wiki/Well-known_text_representation_of_geometry

Extracting Vector Layers from OpenStreetMap




```
$ ogrinfo map.osm # 122 KB of uncompressed XML

1: points (Point)
2: lines (Line String)
3: multilinestrings (Multi Line String)
4: multipolygons (Multi Polygon)
5: other_relations (Geometry Collection)
```

```
$ for LAYER in points \
               lines \
               multipolygons; do
        echo ".mode line
              SELECT other_tags,
                     wkb_geometry wkt
              FROM st_read('map.osm', layer='$LAYER')
              LIMIT 1;" \
            | duckdb -unsigned
  done


other_tags = "crossing"=>"uncontrolled", "crossing:markings"=>"zebra"
       wkt = POINT (24.7356999 59.4348117)

other_tags = …"surface"=>"cobblestone", "tunnel"=>"building_passage"
       wkt = LINESTRING (24.7377793 59.4356907, 24.7377398 59.4358396)

other_tags =
       wkt = MULTIPOLYGON (((24.7377173 59.4373743,
                             24.7371252 59.4374078,
                             24.7366801 59.4369198,...)))
```

Open Geospatial Consortium: Standards

- 3D Tiles
- Catalog Service for the Web
- Geography Markup Language
- GeoPackage
- GeoParquet
- GeoSPARQL
- GeoXACML
- Keyhole Markup Language
- Observations and Measurements
- OGC Reference Model
- Open Location Service
- OGC Web Service Common
- Sensor Observation Service
- Sensor Planning Service
- SensorML
- SensorThings
- Simple Features (SQL)
- Styled Layer Descriptor
- SRID, an identification for spatial coordinate systems
- WaterML
- Web Coverage Service
- Web Coverage Processing Service
- Web Feature Service
- Web Map Service
- Web Map Tile Service
- Web Processing Service
- Web Terrain Service

OGC 06-104r4: Simple Feature Access

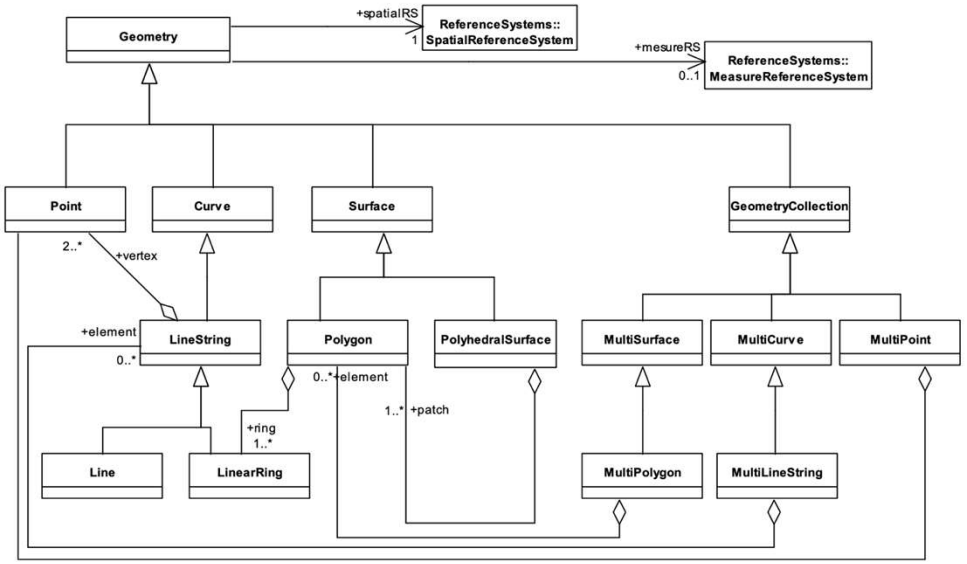


Figure 4: Figure: SQL Geometry Type hierarchy

The SQL/MM `ST_Dimension`, `ST_GeometryType`, `ST_AsText`, `ST_AsBinary`, `ST_SRID`, `ST_IsEmpty`, `ST_IsSimple`, `ST_Boundary`, and `ST_Envelope` routines shall be supported for all Geometry Types. Also included are SQL routines for obtaining the Well-known Binary and Text Representation of a geometric object and creating values from them.

Consistent with the definitions of relations in Part 1, Clause 6.1.2.3, the SQL/MM `ST_Equals`, `ST_Disjoint`, `ST_Intersects`, `ST_Touches`, `ST_Crosses`, `ST_Within`, `ST_Contains`, `ST_Overlaps` and `ST_Relate` routines shall be supported to test named spatial relationships between two geometric objects.

5.1 Abbreviations

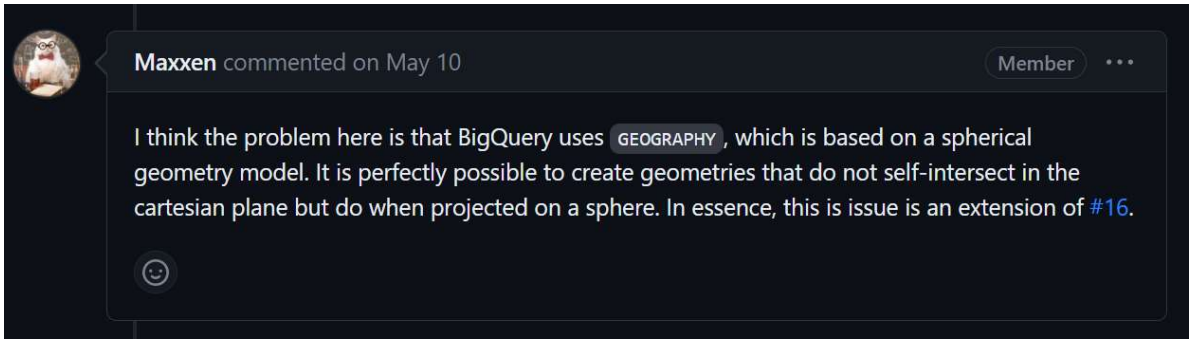
FID	Feature ID column in the implementation of feature tables based on predefined data types
GID	Geometry ID column in the implementation of feature tables based on predefined data types
MM	Multimedia
SQL	Structured query language, not an acronym, pronounced as "sequel"
SQL/MM	SQL Multimedia and Application Packages
SRID	Spatial Reference System Identifier
SRTEXT	Spatial Reference System Well Known Text
WKB	Well-Known Binary (representation for example, geometry)
WKT	Well-Known Text
WKTR	Well-Known Text Representation

https://www.ogc.org/standard/sfs/

Invalid Geometry

```
$ duckdb -unsigned

SELECT ST_IsValid(
        ST_GEOMFROMTEXT('MULTIPOLYGON(((-122.40246
                37.724549,...)))'))
            AS geom_valid;

┌──────────────┐
│ geom_valid   │
│  boolean     │
├──────────────┤
│ true         │
└──────────────┘


$ echo "…C1995EC0BDAC8905BEDC4240" > bad_poly.csv

$ bq load --source_format=CSV \
    --quiet \
    geo.wkb_test \
    ./bad_poly.csv \
    geom:GEOGRAPHY
```

```
"Loop  edge  crosses loop  edge ; in WKB geography"
"Invalid polygon loop: Edge  crosses edge ; in WKB geography"
"Loop  edge  crosses loop  edge ; in polygon ; in WKB geography"
"Invalid nesting: loop  should not contain loop ; in WKB geography"
"Invalid polygon loop: Edge  crosses edge ; in loop ; in WKB geography"
"Invalid polygon loop: Edge  crosses edge ; in polygon ; in WKB geography"
"Invalid polygon loop: Edge  has duplicate vertex with edge ; in WKB geography"
"Invalid nesting: loop  should not contain loop ; in polygon ; in WKB geography"
"Invalid polygon loop: Edge  crosses edge ; in loop ; in polygon ; in WKB geography"
"Invalid polygon loop: Edge  has duplicate vertex with edge ; in loop ; in WKB geography"
"Invalid polygon loop: Edge  has duplicate vertex with edge ; in polygon ; in WKB geography"
"Polygon loop should have at least  unique vertices, but only had ; in loop ; in WKB geograp
"Polygon loop should have at least  unique vertices, but only had ; in polygon ; in WKB geog
"Invalid polygon loop: Edge  has duplicate vertex with edge ; in loop ; in polygon ; in WKB
```



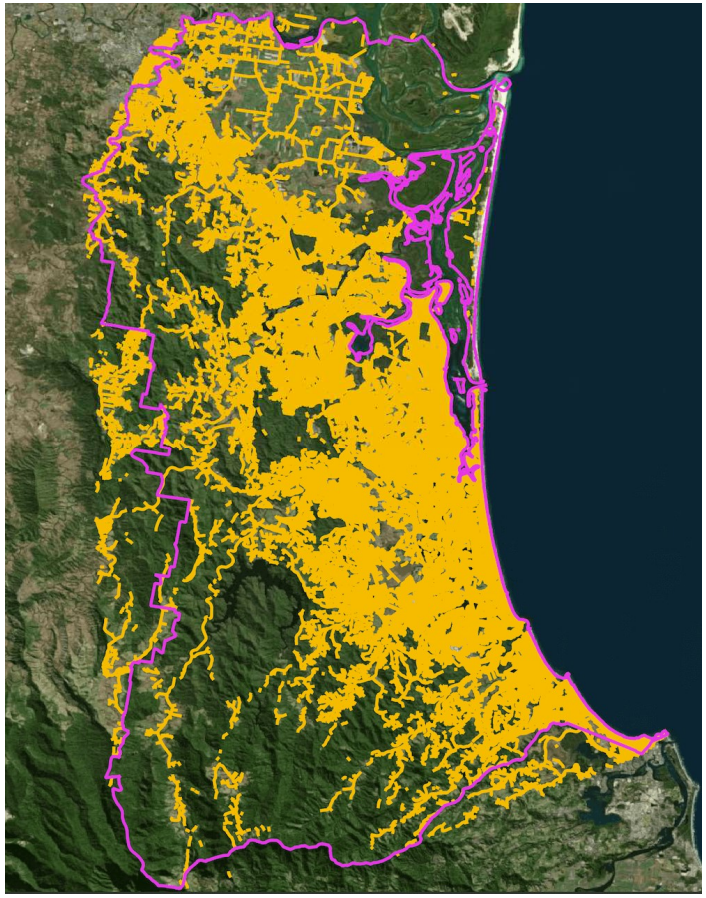
https://github.com/duckdblabs/duckdb_spatial/issues/73

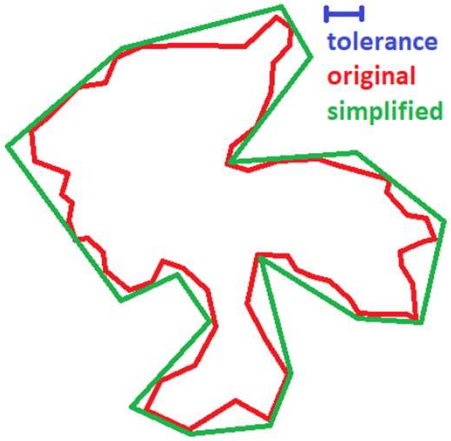
Invalid Geometry: BigQuery's make_valid

```
-- In PostGIS
COPY
(SELECT TRANSLATE(
                  ENCODE((ST_ASGEOJSON("geom")::JSONB
                          - 'crs')::TEXT::BYTEA,
                         'base64'),
                  E'\n',
                  '') geom_,
        "h3_7",
        "h3_8",
        "h3_9"
 FROM "addresses")
TO PROGRAM 'split --line-bytes=4000000000 --filter="pigz -1 > /home/mark/tmp123/\$FILE.csv.gz"' WITH CSV;
```

```
-- In BigQuery:

ALTER TABLE dataset.addresses
ADD COLUMN geom GEOGRAPHY;

UPDATE dataset.addresses
SET geom = ST_GEOGFROMGEOJSON(
                SAFE_CONVERT_BYTES_TO_STRING(FROM_BASE64(geom_)),
                make_valid => True)
WHERE True;
```

Convex Hulls & Geometry Simplification





ST_CONVEXHULL

```
ST_CONVEXHULL(geography_expression)
```

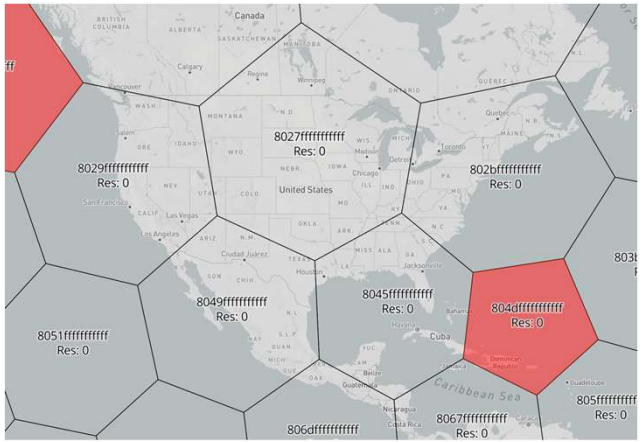
Description

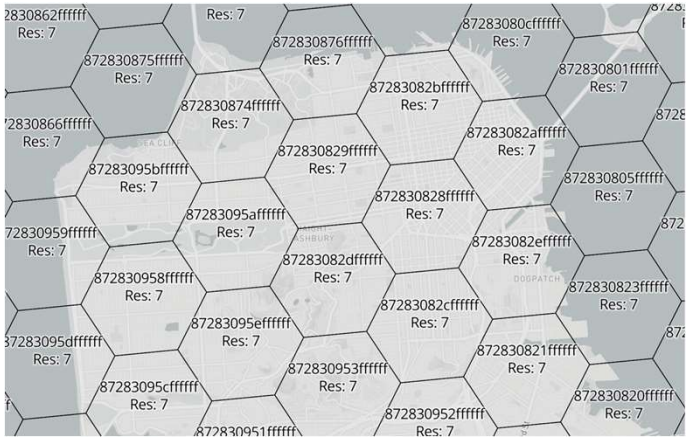
Returns the convex hull for the input `GEOGRAPHY`. The convex hull is the smallest convex `GEOGRAPHY` that covers the input. A `GEOGRAPHY` is convex if for every pair of points in the `GEOGRAPHY`, the geodesic edge connecting the points are also contained in the same `GEOGRAPHY`.

```python
import json
from shapely import wkb, wkt
from shapely.geometry import shape


def unique(sequence):
    seen = set()
    return [x for x in sequence if not (x in seen or seen.add(x))]


with open('out.csv', 'w') as f:
    for num, line in enumerate(open('in.geojson')):
        try:
            geom = shape(json.loads(line)['geometry'])
        except Exception:
            continue

        geom = wkt.dumps(geom, rounding_precision=6)
        points = unique([x.strip()
                         for x in str(geom)
                                    .split('(')[-1]
                                    .split(')')[0]
                                    .split(',')])
        if len(points) < 2:
            continue

        f.write('"LINESTRING (%s)"\n' % ','.join(points))
```

```
LINESTRING (-115.002877877877 47.181808877877,
            -115.002362362362 47.182078362362,
            -115.002362362362 47.182078362362,
            -115.002136136146 47.182056136146,
            -115.001879879879 47.182093879879,
            -115.001535535535 47.182122535535)

LINESTRING (-115.002877 47.181808,
            -115.002362 47.182078,
            -115.002136 47.182056,
            -115.001879 47.182093,
            -115.001535 47.182122)
```

H3: Uber’s Hexagonal Hierarchical Spatial Grid



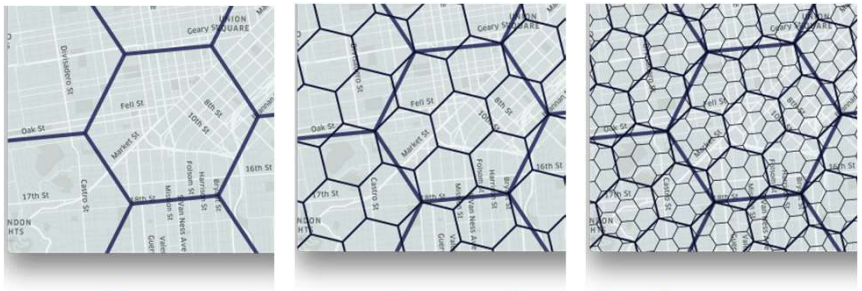


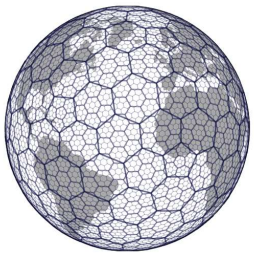

```
$ psql geodata
```

```
\x on
SELECT *
FROM microsoft_buildings
LIMIT 1;
```

```
revision   | 2
from_date  | 2019-07-17
to_date    | 2019-09-25
geom       | 0102000000050000000F148BC3
h3_7       | 872656731ffffff
h3_8       | 8826567311fffff
h3_9       | 8926567311bffff
```

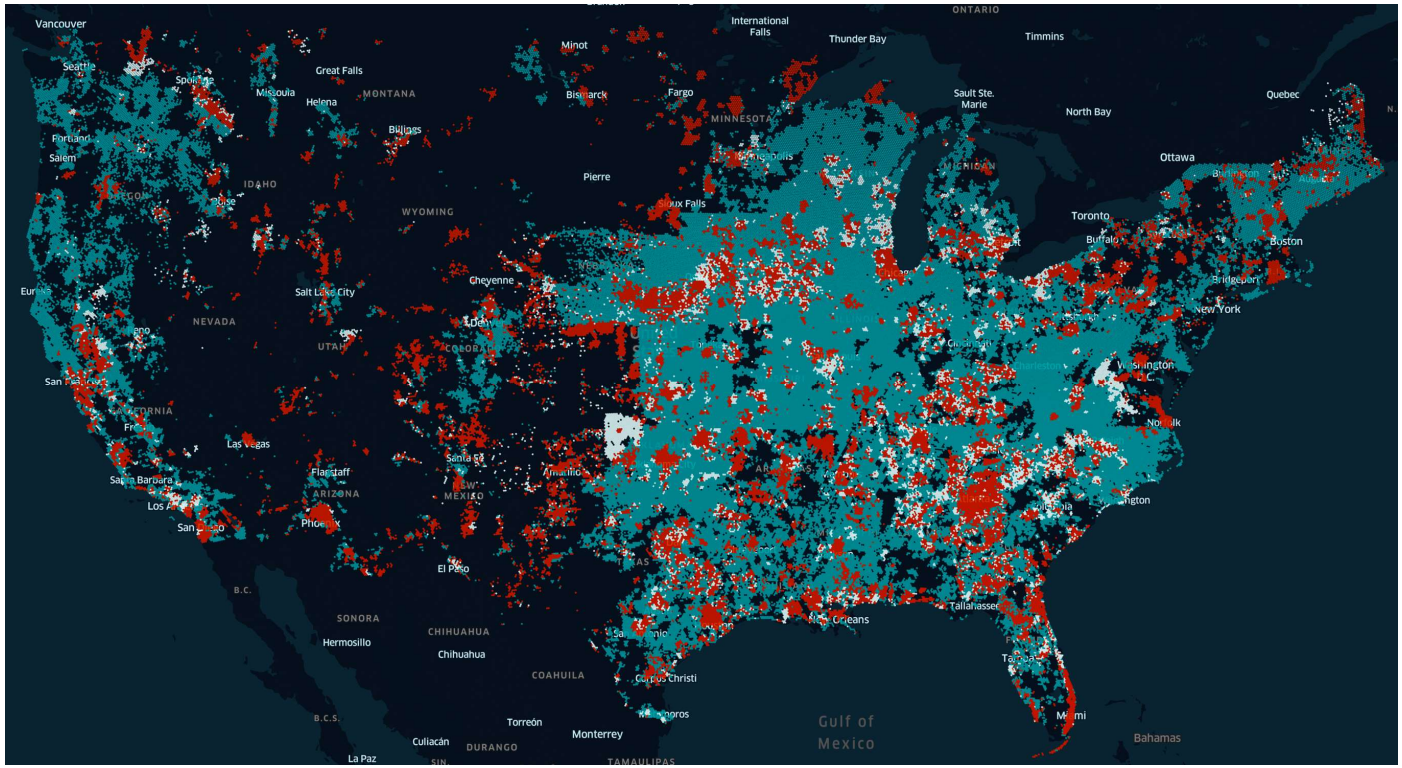
Res	Average Hexagon Area (km^2)
0	4,357,449.416078381
1	609,788.441794133
2	86,801.780398997
3	12,393.434655088
4	1,770.347654491
5	252.903858182
6	36.129062164
7	5.161293360
8	0.737327598
9	0.105332513

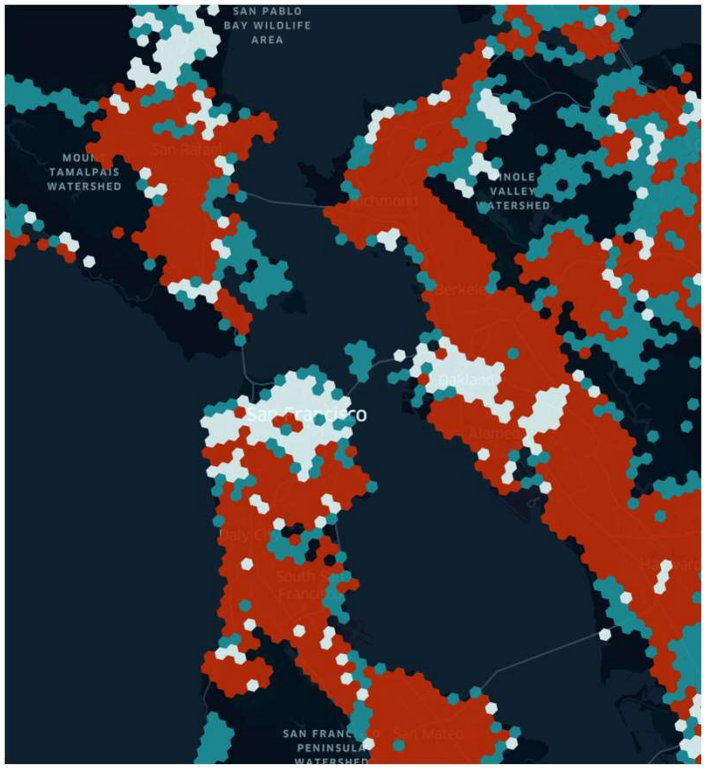




https://www.uber.com/en-EE/blog/h3/
https://h3geo.org/docs/core-library/restable/
https://tech.marksblogg.com/faster-geospatial-enrichment.html
https://what-the-h3index.vercel.app/

H3s with Unfolded



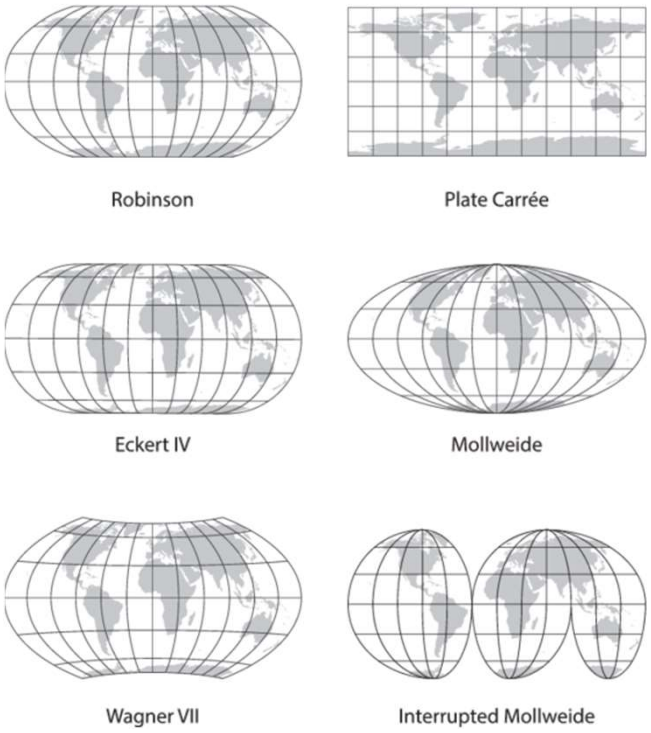


https://tech.marksblogg.com/duckdb-geospatial-gis.html

H3s with QGIS

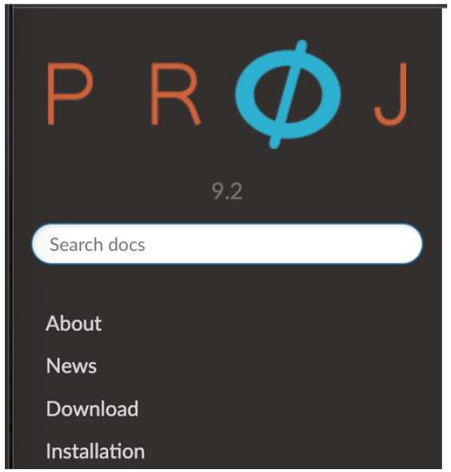
https://tech.marksblogg.com/making-heatmaps-python-qgis.html
https://tech.marksblogg.com/duckdb-gis-spatial-extension.html

Projection: Distorting a lumpy sphere



European Petroleum Survey Group

EPSG



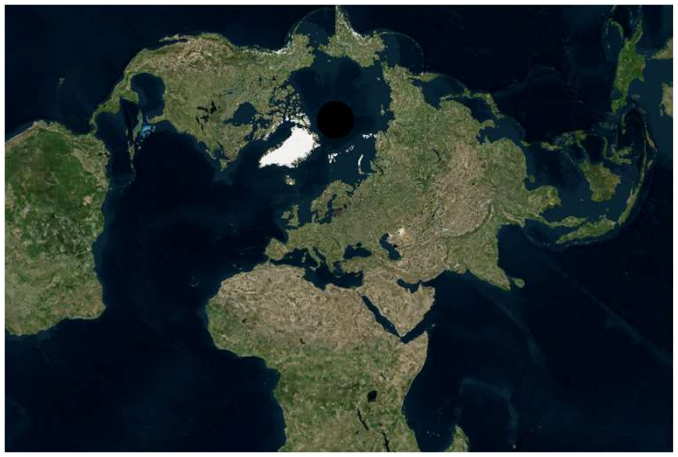
> … from time to time Esri also receives coordinate system-related information that has not been published in the EPSG dataset directly from organizations, government agencies, or companies. In those cases, Esri will include that information in our projection utilities. The 'authority' for the object is 'ESRI', and the associated code number is assigned by Esri, not EPSG.

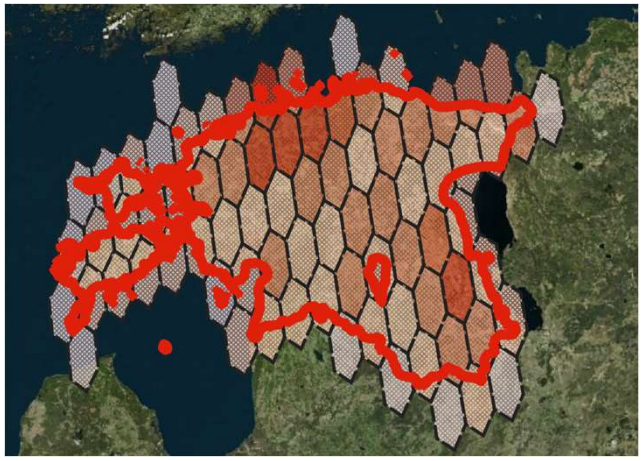
https://en.wikipedia.org/wiki/PROJ
https://en.wikipedia.org/wiki/Map_projection
https://pubs.er.usgs.gov/publication/pp1395
https://support.esri.com/en-us/knowledge-base/faq-what-does-authority-epsg-mean-in-an-arcgis-desktop-000011199

Projection: EPSG:4326 vs EPSG:3300

EPSG:4326

EPSG:3300





Projection: EPSG:4326 vs EPSG:3300

WGS 84

Properties

- Geographic (uses latitude and longitude for coordinates)
- Dynamic (relies on a datum which is not plate-fixed)
- Celestial body: Earth
- Based on *World Geodetic System 1984 ensemble* (EPSG:6326), which has a limited accuracy of **at best 2 meters**.
- Method: Lat/long (Geodetic alias)

WKT

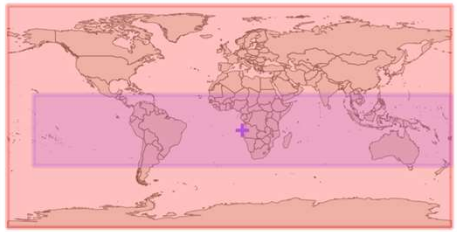
```
GEOGCRS["WGS 84",
    ENSEMBLE["World Geodetic System 1984 ensemble",
        MEMBER["World Geodetic System 1984 (Transit)"],
        MEMBER["World Geodetic System 1984 (G730)"],
        MEMBER["World Geodetic System 1984 (G873)"],
        MEMBER["World Geodetic System 1984 (G1150)"],
        MEMBER["World Geodetic System 1984 (G1674)"],
        MEMBER["World Geodetic System 1984 (G1762)"],
        ELLIPSOID["WGS 84",6378137,298.257223563,
            LENGTHUNIT["metre",1]],
        ENSEMBLEACCURACY[2.0]],
    PRIMEM["Greenwich",0,
        ANGLEUNIT["degree",0.0174532925199433]],
    CS[ellipsoidal,2],
        AXIS["geodetic latitude (Lat)",north,
            ORDER[1],
            ANGLEUNIT["degree",0.0174532925199433]],
        AXIS["geodetic longitude (Lon)",east,
            ORDER[2],
            ANGLEUNIT["degree",0.0174532925199433]],
    USAGE[
        SCOPE["Horizontal component of 3D system."],
        AREA["World."],
        BBOX[-90,-180,90,180]],
    ID["EPSG",4326]]
```

Proj4

```
+proj=longlat +datum=WGS84 +no_defs
```

Extent

```
-180.00,-90.00,180.00,90.00
```



Estonian Coordinate System of 1992

Properties

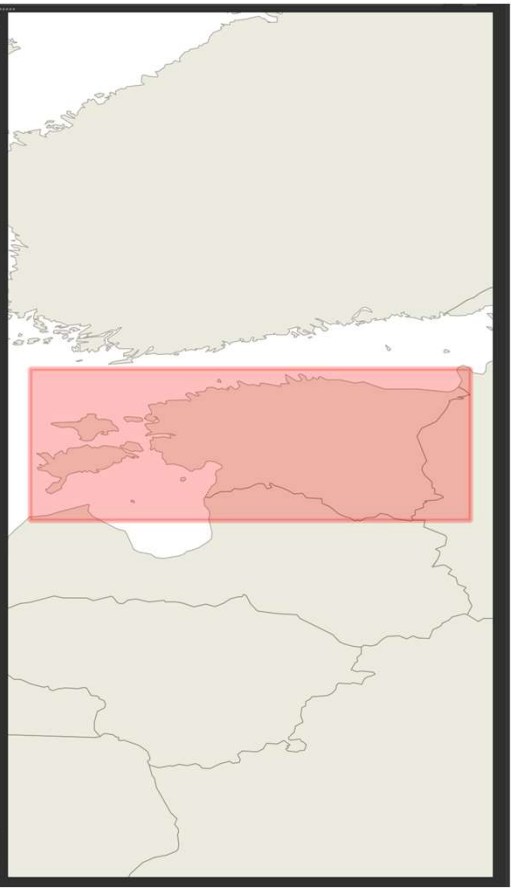
- Units: meters
- Static (relies on a datum which is plate-fixed)
- Celestial body: Earth
- Method: Lambert Conformal Conic

WKT

```
PROJCRS["Estonian Coordinate System of 1992",
    BASEGEOGCRS["EST92",
        DATUM["Estonia 1992",
            ELLIPSOID["GRS 1980",6378137,298.257222101,
                LENGTHUNIT["metre",1]]],
        PRIMEM["Greenwich",0,
            ANGLEUNIT["degree",0.0174532925199433]],
        ID["EPSG",4133]],
    CONVERSION["Estonian National Grid",
        METHOD["Lambert Conic Conformal (2SP)",
            ID["EPSG",9802]],
        PARAMETER["Latitude of false origin",
57.5175539305556,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8821]],
        PARAMETER["Longitude of false origin",24,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8822]],
        PARAMETER["Latitude of 1st standard parallel",
59.3333333333333,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8823]],
        PARAMETER["Latitude of 2nd standard parallel",58,
            ANGLEUNIT["degree",0.0174532925199433],
            ID["EPSG",8824]],
        PARAMETER["Easting at false origin",500000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8826]],
        PARAMETER["Northing at false origin",6375000,
            LENGTHUNIT["metre",1],
            ID["EPSG",8827]]],
    CS[Cartesian,2],
        AXIS["northing (X)",north,
            ORDER[1],
            LENGTHUNIT["metre",1]],
        AXIS["easting (Y)",east,
            ORDER[2],
            LENGTHUNIT["metre",1]],
    USAGE[
        SCOPE["Topographic mapping (large scale)."],
        AREA["Estonia - onshore."],
        BBOX[57.52,21.74,59.75,28.2]],
    ID["EPSG",3300]]
```

Proj4

```
+proj=lcc +lat_0=57.5175539305556 +lon_0=24
+lat_1=59.3333333333333 +lat_2=58 +x_0=500000
+y_0=6375000 +ellps=GRS80
+towgs84=0.055,-0.541,-0.185,0.0183,-0.0003,-0.007,-0.014
+units=m +no_defs
```



Intermixing WGS84 and EPSG:4236

4326 is just the EPSG identifier of WGS84.

WGS84 comprises a standard coordinate frame for the Earth, a datum/reference ellipsoid for raw altitude data.

7 However GeoJSON coordinate system is WGS84 but not EPSG:4326, (there's a difference in axis order) so whilst closely related, they should not be used as synonymous terms. – nmtoken Jun 18, 2021 at 5:43

If you're really going to pick a nit: EPSG 4326 defines a full coordinate reference system, providing spatial meaning to otherwise meaningless pairs of numbers. It means "latitude and longitude coordinates on the WGS84 reference ellipsoid."

The term WGS84 is sometimes used the same way, but also it can refer to the ellipsoid only. For example, you can have "meters northing and easting as measured upon the cylinder formed by projecting the WGS84 ellipsoid using a transverse mercator projection with a central meridian of -123 degrees". (http://spatialreference.org/ref/epsg/32610/)

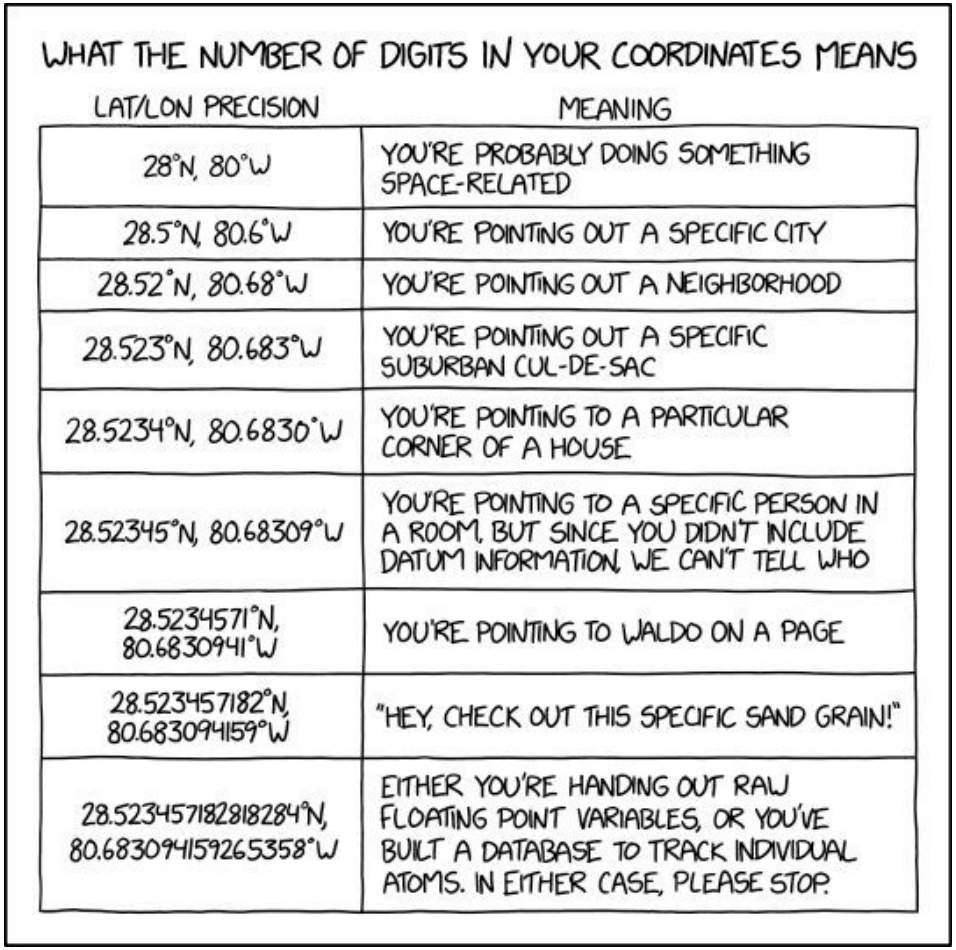
In any case: No difference, just like everyone else is saying, except in the very nitty gritty details of how they are used.

TLDR: No one says "the EPSG 4326 ellipsoid" as part of a coordinate system definition.

QGIS gives WGS84 as the Coordinate Reference System and EPSG:4236 as the Authority ID.

Packaging GIS: CSV Files

```
$ ls -lh locations.csv.gz # 1.8 GB

$ echo "COPY (SELECT *
              FROM 'locations.csv.gz'
              ORDER BY longitude)
         TO 'locations.sorted.csv';" \
    | duckdb -unsigned sorting.duckdb

$ pigz -9  locations.sorted.csv
$ ls   -lh locations.sorted.csv.gz # 1.3 GB
```



WHAT THE NUMBER OF DIGITS IN YOUR COORDINATES MEANS

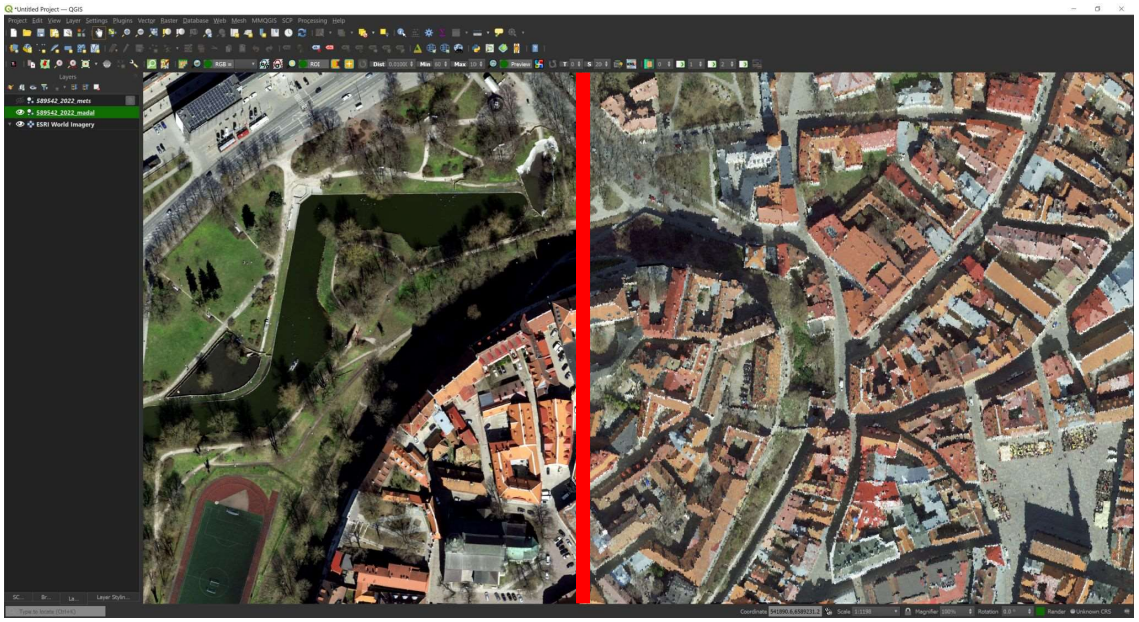
LAT/LON PRECISION	MEANING
28°N, 80°W	YOU'RE PROBABLY DOING SOMETHING SPACE-RELATED
28.5°N, 80.6°W	YOU'RE POINTING OUT A SPECIFIC CITY
28.52°N, 80.68°W	YOU'RE POINTING OUT A NEIGHBORHOOD
28.523°N, 80.683°W	YOU'RE POINTING OUT A SPECIFIC SUBURBAN CUL-DE-SAC
28.5234°N, 80.6830°W	YOU'RE POINTING TO A PARTICULAR CORNER OF A HOUSE
28.52345°N, 80.68309°W	YOU'RE POINTING TO A SPECIFIC PERSON IN A ROOM. BUT SINCE YOU DIDN'T INCLUDE DATUM INFORMATION, WE CAN'T TELL WHO
28.5234571°N, 80.6830941°W	YOU'RE POINTING TO WALDO ON A PAGE
28.523457182°N, 80.683094159°W	"HEY, CHECK OUT THIS SPECIFIC SAND GRAIN!"
28.523457182818284°N, 80.683094159265358°W	EITHER YOU'RE HANDING OUT RAW FLOATING POINT VARIABLES, OR YOU'VE BUILT A DATABASE TO TRACK INDIVIDUAL ATOMS. IN EITHER CASE, PLEASE STOP.

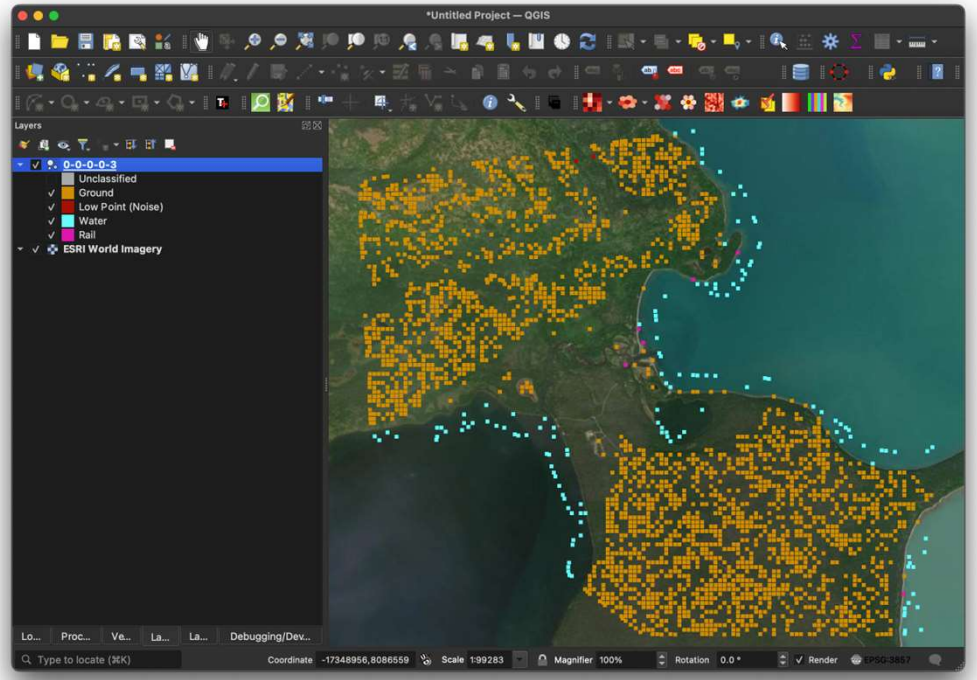
https://xkcd.com/2170/

Packaging GIS: Line-delimited JSON

```
# 7,803,427
$ wc -c country.json.gz

# 4,668,688 bytes, 1.67x smaller
# no white spaces between keys and values
$ gunzip -c country.json.gz \
    | jq -Sc \
    | sort \
    | gzip -9 \
    | wc -c

# 4,642,056 bytes, 26,632 bytes smaller
$ gunzip -c country.json.gz \
    | jq -Sc \
    | sort \
    | pigz -9 \
    | wc -c
```

LiDAR



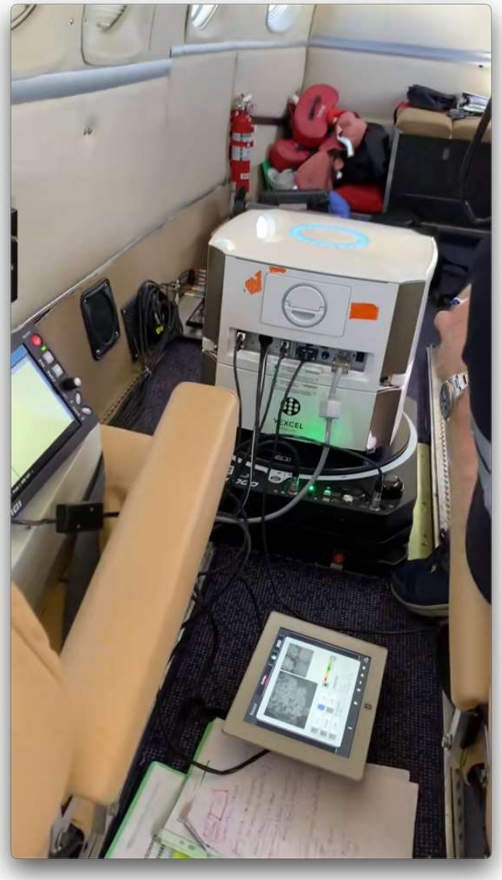


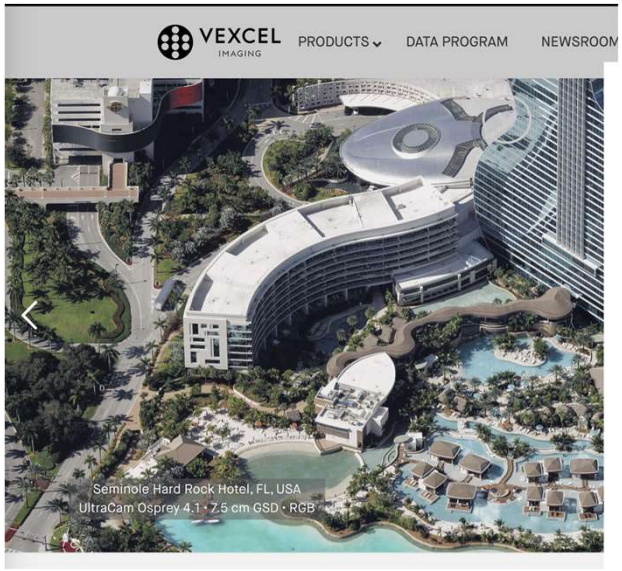
https://pdal.io/en/latest/
https://geoportaal.maaamet.ee/eng/Spatial-Data-p58.html

USGS LiDAR Data on AWS

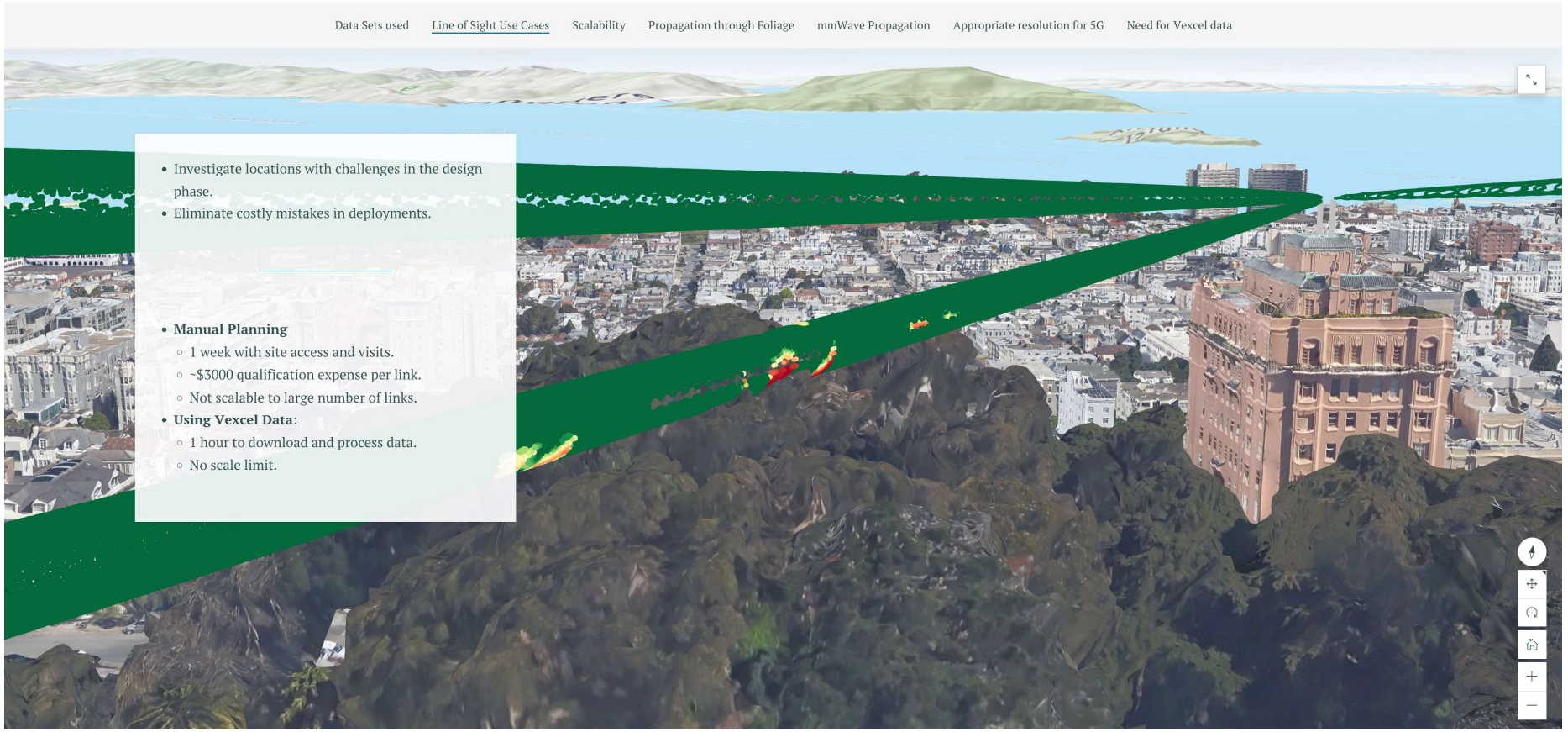
```
$ aws s3 ls --no-sign-request \
    s3://usgs-lidar-public/ \
    > datasets.txt # 1,991


$ aws s3 ls --no-sign-request \
    s3://usgs-lidar-public/ \
    --recursive \
    --summarize \
    > list.txt # 106 GB of file listing


$ cat list.txt \
    | cut -c20-31 \
    | awk '{ sum += $1 } END { print sum }' # ~300 TB

$ aws s3 cp \
    --no-sign-request \
    s3://usgs-lidar-public/AK_BrooksCamp_2012/ept-data/0-0-0-0-3.laz \
    ./
```

LiDAR Data Captured with Vexcel





LiDAR: Detecting Trees & Line-of-sight Issues



https://storymaps.arcgis.com/stories/65566e6ac1c54aa18f16e5a1ce3e06e6

OpenStreetMap (OSM)

Nearby features

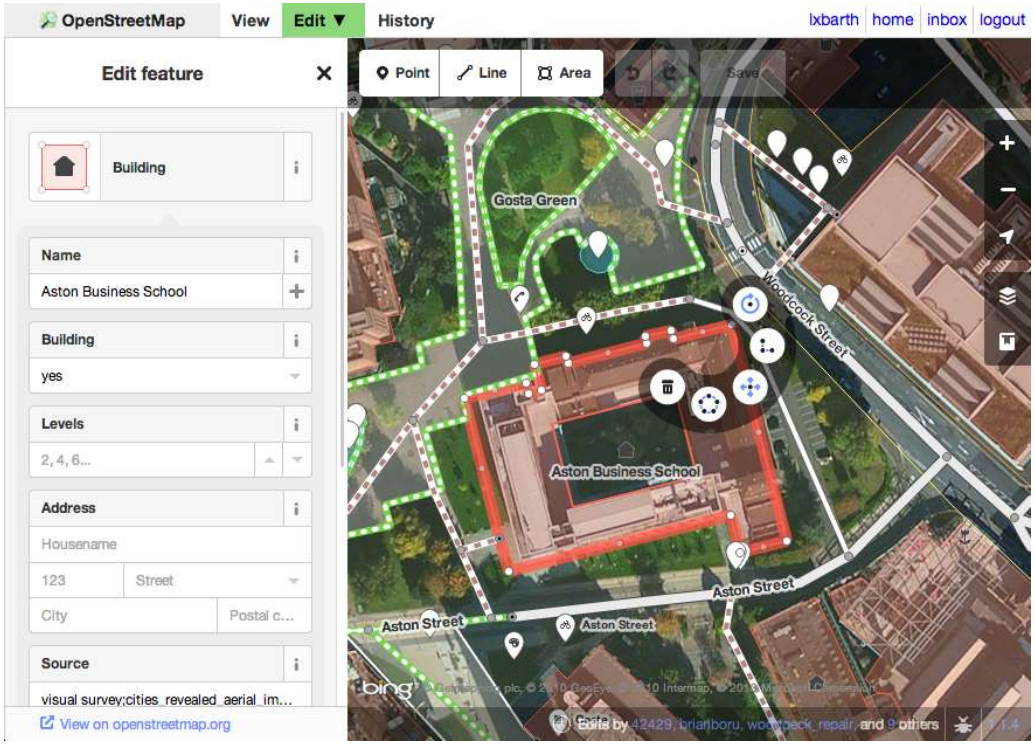
- Crossing #1800775442
- Cycle Rental #6643150117
- Tree #7862067056
- Information Parcours Princesse Grace - 04
- Pedestrian Way Place d'Armes
- Secondary Road Avenue du Port
- Footpath #168897635
- Secondary Road Avenue du Port
- Suburb Boundary #398362855
- Residential Road Avenue du Port
- Tree Row #572933761
- Pipeline #722801035
- Building #833504233
- Footpath #850230911
- Way #905455474
- Relation Bus 1: Monaco Ville (le Rocher) → Saint-Roman
- Relation Bus 2 : Monaco Ville (le Rocher) => Jardin Exotique
- Relation Bus 4: Saint-Roman → Fontvieille - Centre Commercial

```
way_id | 257071645
type   | residential
name   | Avenue Jacques Abba
tags   | {
       |     "name": "Avenue Jacques Abba",
       |     "source": "cadastre-dgi-fr source : Di
       |     "highway": "residential"
       | }
geom   | SRID=3857;LINESTRING,(
       | 824897.887300565 5423604.430898497,
       | 824887.1894974997 5423599.5012846105,
       | 824877.5826254443 5423593.970877169,
       | 824869.3895109219 5423586.822945994,
       | 824862.8661887613 5423578.088304859,
       | 824825.1511452807 5423522.060477127,
       | 824816.4125652533 5423496.488361983,
       | 824814.9097521277 5423478.896020303,
       | 824812.1156329087 5423432.127050894,
       | 824807.6183254807 5423384.511061079,
       | 824804.4123241458 5423365.13199355,
       | 824801.5848090796 5423355.350047257,
       | 824797.0986336006 5423347.201000705,
       | 824791.8220897372 5423339.930023846,
       | 824785.3210314749 5423331.38047152,
       | 824728.0248895634 5423273.567220179,
       | 824714.2658005015 5423260.011274419,
       | 824706.9075821601 5423250.09079946,
       | 824701.8202814307 5423239.461730986,
       | 824693.0594375054 5423223.964879177,
       | 824685.2559412007 5423208.806949899,
       | 824681.3263631756 5423202.429535876,
       | 824677.2186739653 5423197.885245964,
       | 824675.2149231312 5423195.682421491,
       | 824656.0234429184 5423179.184361206,
       | 824642.7096318195 5423161.130495241,
       | 824637.5555393959 5423148.961100397,
       | 824631.7891897728 5423113.531308058)
```

NOT 4326!

https://tech.marksblogg.com/streets-of-monaco-openstreetmap-postgis-qgis.html

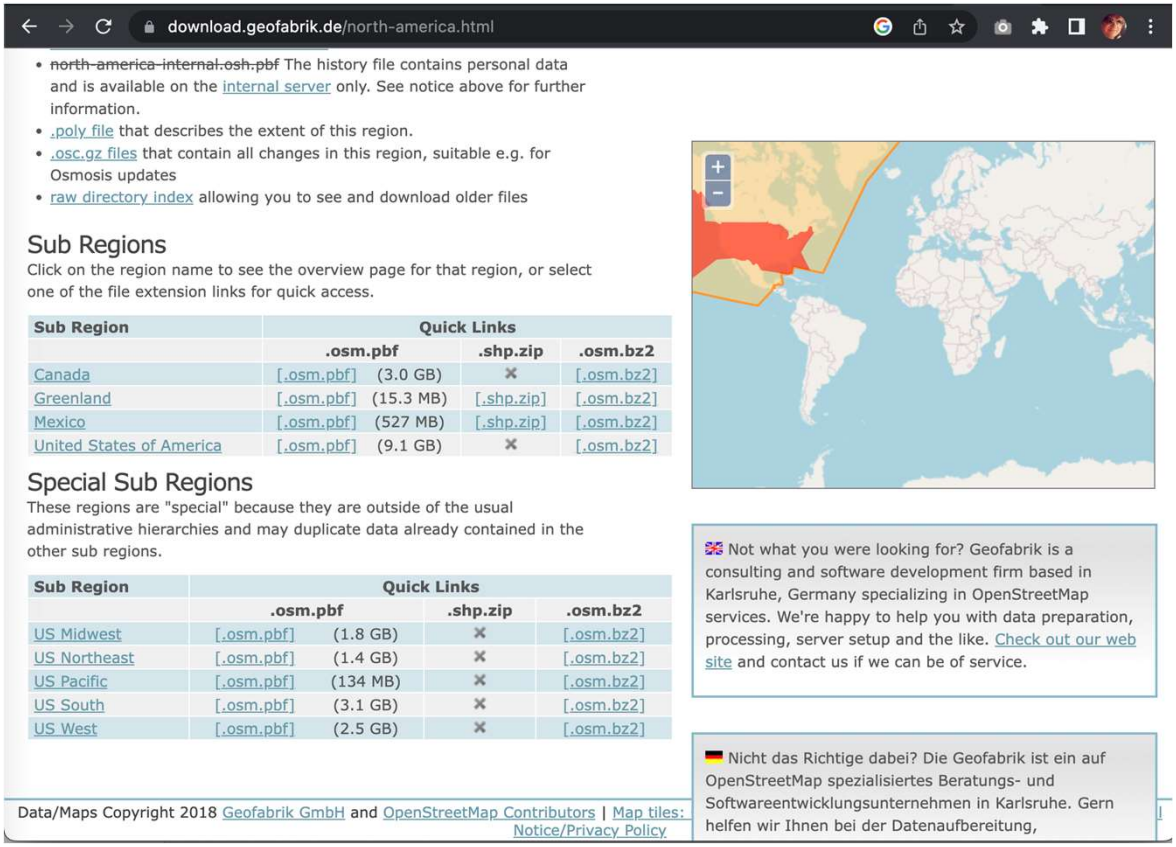
OpenStreetMap Editor



highway	trunk		The most important roads in a country's system that aren't motorways. (Need not necessarily be a divided highway.)		
highway	primary		The next most important roads in a country's system. (Often link larger towns.)		
highway	secondary		The next most important roads in a country's system. (Often link towns.)		

https://wiki.openstreetmap.org/wiki/Map_features#Highway
https://learnosm.org/en/beginner/id-editor/

OpenStreetMap Extracts from GeoFabrik

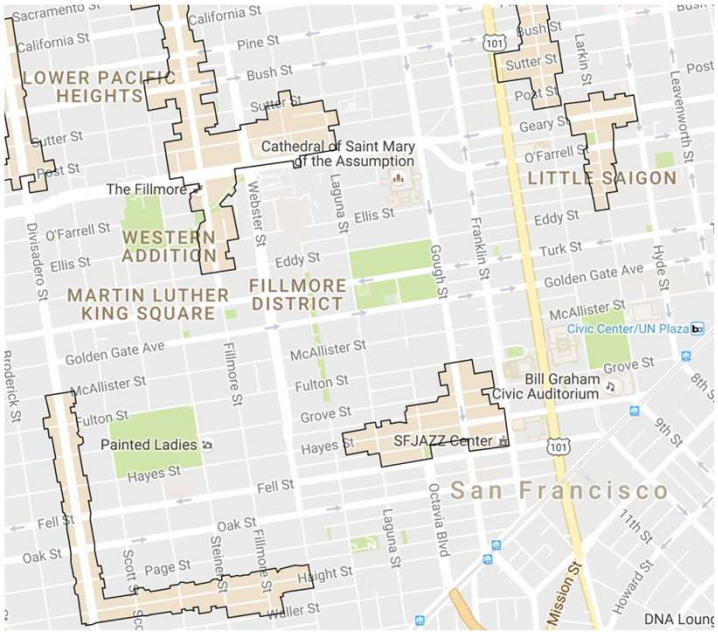


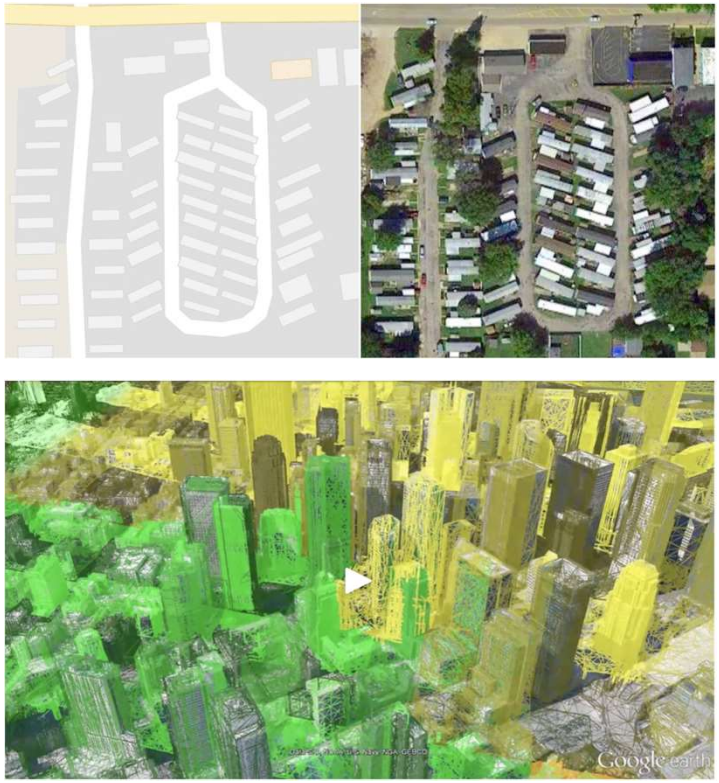
Read with:

- DuckDB (via Spatial extension & GDAL)
- osm2pgsql
- osmium-tool
- PostGIS
- QGIS

https://download.geofabrik.de/

Google Maps' Moat

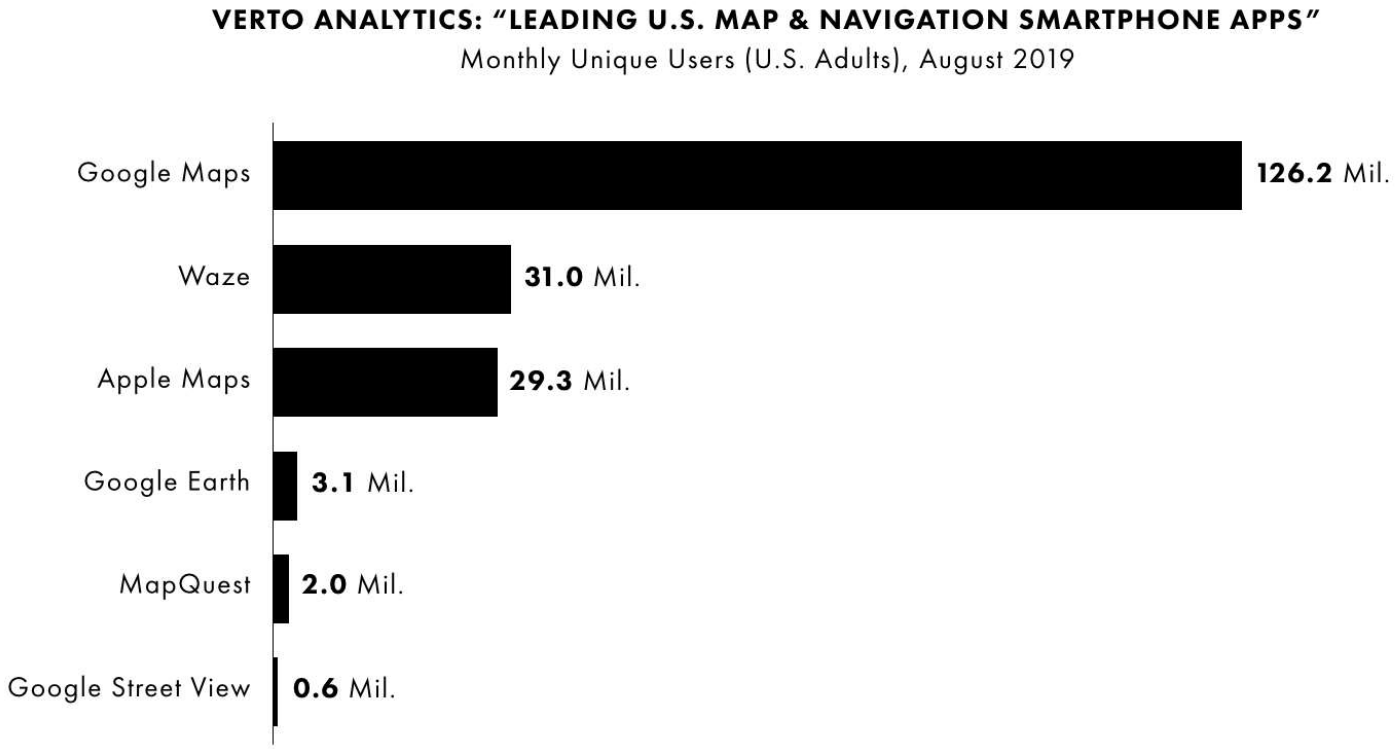




- 3D Buildings, even in small towns
- Satellites on buildings
- Trailers and sheds
- Areas of interest (~23 major shopping streets in San Fran)
- Door locations on buildings
- Refined label placement

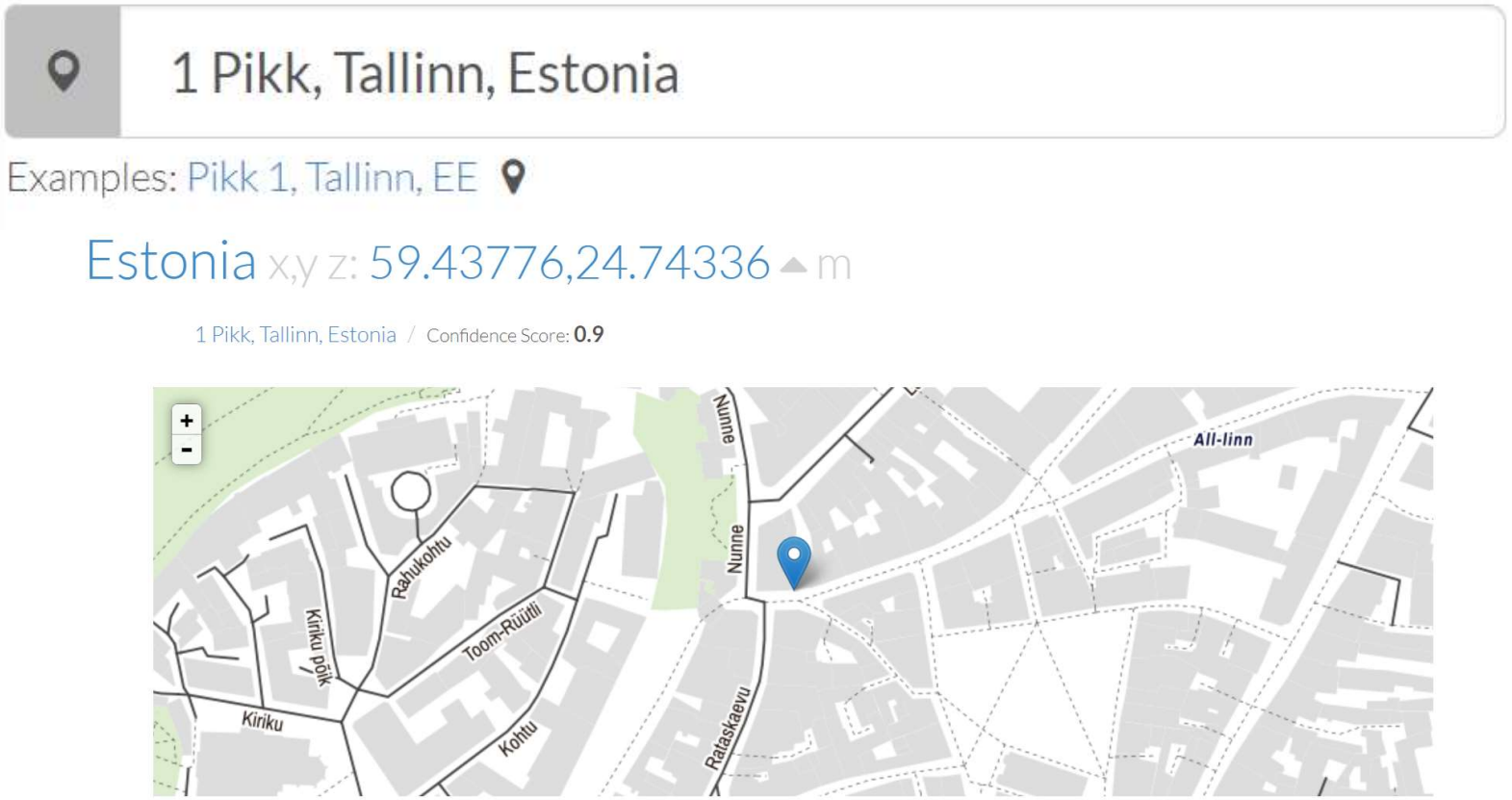
https://www.justinobeirne.com/google-maps-moat

Google Maps' Moat

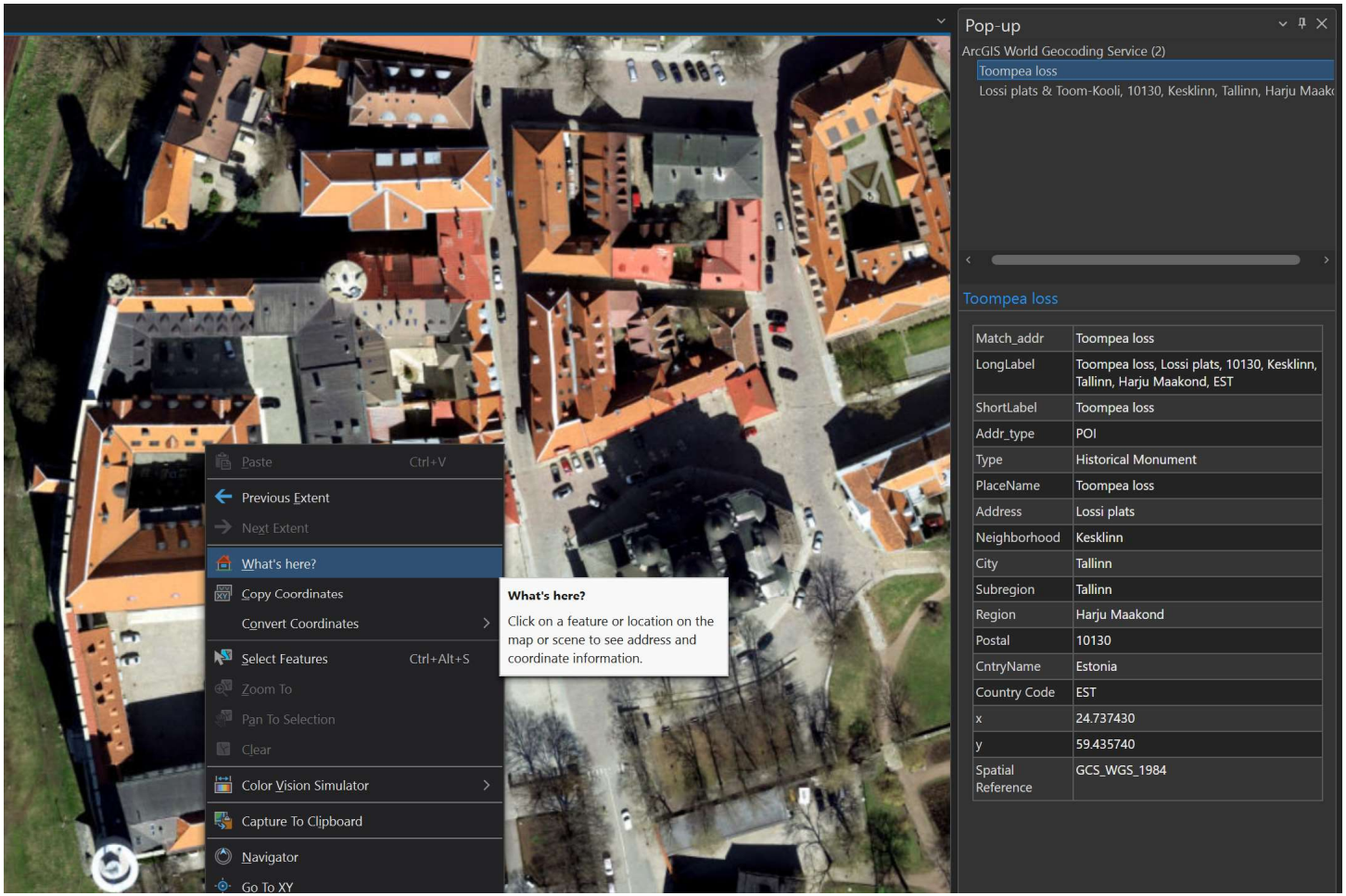


https://www.justinobeirne.com/how-many-people-use-google-maps-compared-to-apple-maps

GIS Applications: Address Geocoding



GIS Applications: Reverse Geocoding



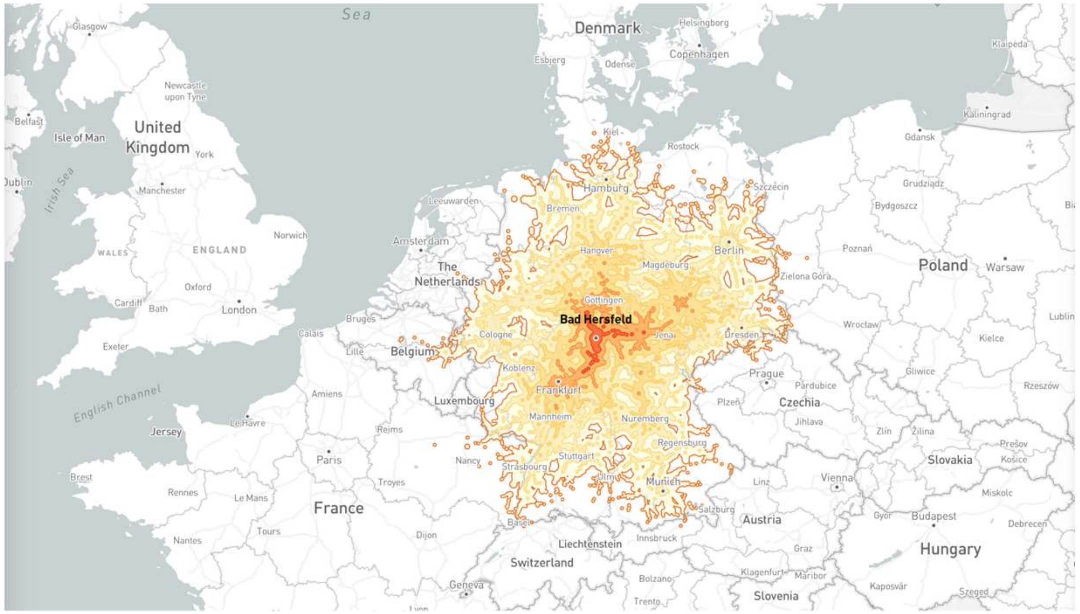
GIS Applications: Route Finding

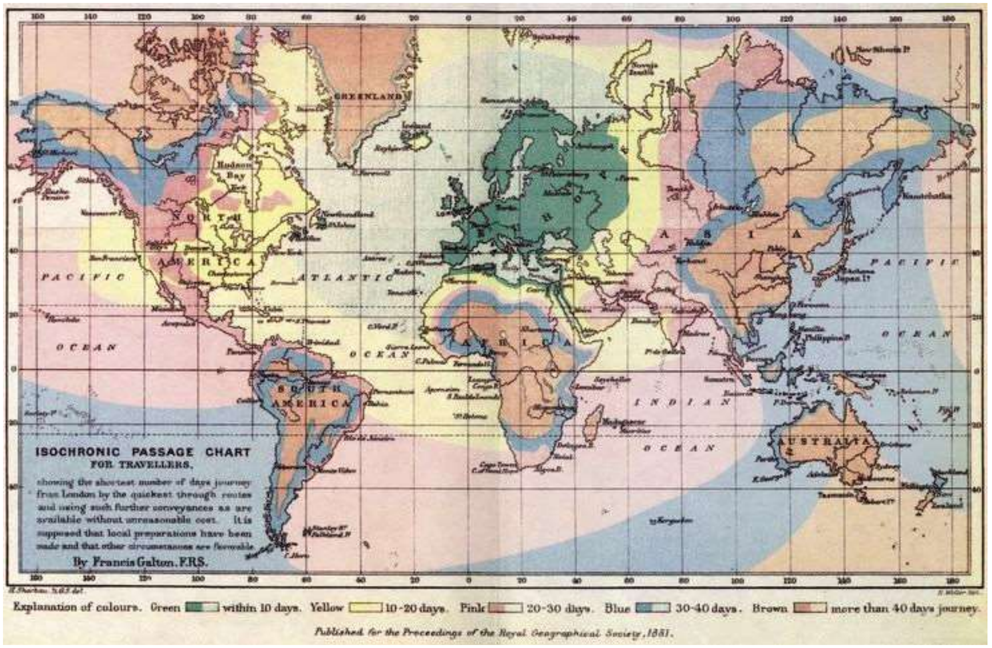
```
SELECT DISTINCT
    stops_pk.stop_name,
    shortest_route.seq                    AS stop_num,
    shortest_route.node                   AS stop_pk,
    ROUND(shortest_route.cost::numeric, 2) AS cost
FROM
    pgr_dijkstra(
        'SELECT trip_pk as id,
                source,
                target,
                distance AS cost
         FROM trip_routes
         WHERE distance IS NOT NULL',
        (SELECT stop_pk
         FROM stops_pk
         WHERE stop_name = 'Vienna Central Station'
         LIMIT 1),
        (SELECT stop_pk
         FROM stops_pk
         WHERE stop_name = 'Oslo (Busterminalen Galleriet)'
         LIMIT 1),
        FALSE
    ) AS shortest_route
LEFT JOIN trip_routes ON shortest_route.edge = trip_routes.trip_pk
LEFT JOIN stops_pk    ON shortest_route.node = stops_pk.stop_pk
ORDER BY shortest_route.seq;
```

stop_name	stop_num	stop_pk	cos
Vienna Central Station	1	753	1.0
Vienna Erdberg (Busterminal VIB)	2	1137	3.3
Prague (ÚAN Florenc bus station)	3	1070	3.7
Berlin central bus station	4	930	4.8
Copenhagen central train station (Ingerslevsgade)	5	1112	4.9
Oslo (Busterminalen Galleriet)	6	1349	0.0

https://tech.marksblogg.com/route-planning-europe-postgresql-pgrouting.html

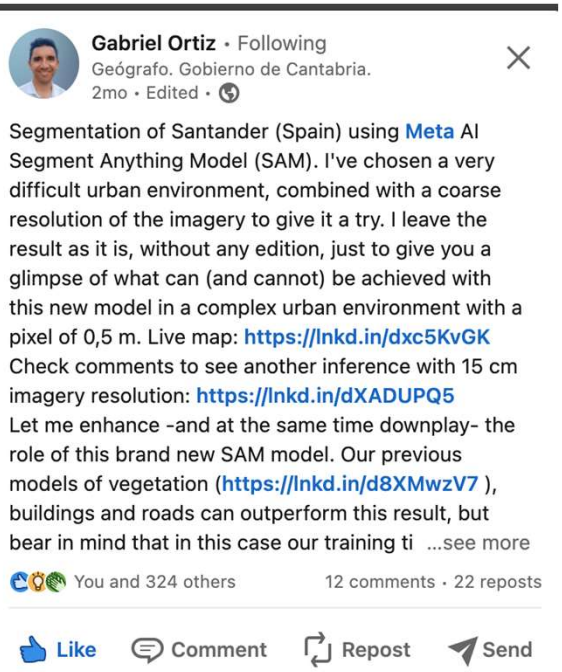
GIS Applications: Isochrones





https://tech.marksblogg.com/valhalla-isochrones.html
https://www.chronotrains.com/

GIS Applications: Segmentation



https://www.youtube.com/watch?v=riirqSyDqMg
https://github.com/microsoft/GlobalMLBuildingFootprints
https://www.linkedin.com/posts/gabriel-ortiz-gis-cantabria_deeplearning-machinelearning-sam-activity-7053797866691059712-eCT3

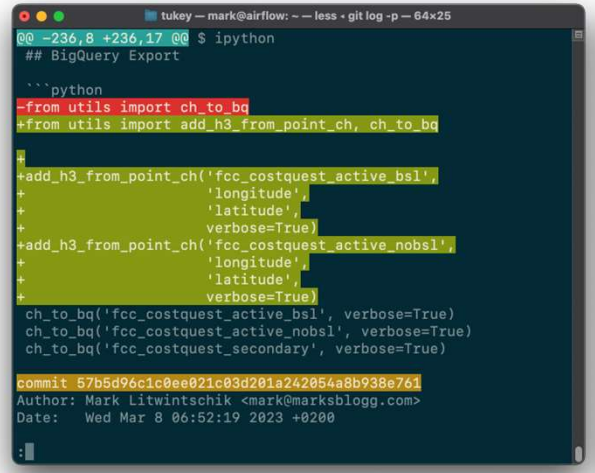
(Mostly) Open Source Software

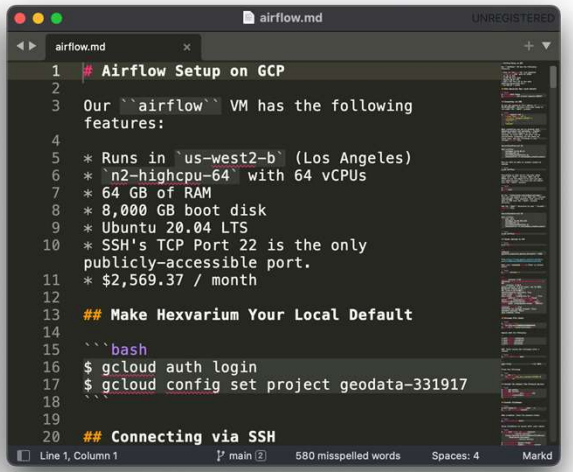
App	Language	LOC (K)	Started	Authors	Desc
GDAL	C++, Python	1,700	1998	Even Rouault, Frank Warmerdam, Kurt Schwehr	Translator for raster and vector GIS files
PROJ	C++	150	1999	Frank Warmerdam, Even Rouault, Kristian Evers	Coordinate transformations
PostGIS	C	200	2001	Sandro Santilli, Regina Obe, Paul Ramsey, Bborie Park	GIS for PostgreSQL
GEOS	C++	150	2002	Sandro Santilli, Paul Ramsey, Daniel Baston, Mateusz Łoskot	Geometry Engine
Shapely	Python, C	19	2007	Sean Gillies, Casper van der Wel, Joris Van den Bossche	GIS tools w/o the need of a DB
Fiona	Python & Cython	15	2007	Sean Gillies, René Buffat	Reads and writes GIS files
QGIS	C++	1,300	2010	Nyall Dawson, Jürgen Fischer, Matthias Kuhn	Photoshop for GIS
GeoPandas	Python	21	2013	Kelsey Jordahl, Joris Van den Bossche, Martin Fleischmann	Python GIS Framework
Valhalla	C++	145	2014	Kevin Kreiser, David Nesbitt, Duane Gearhart	Traffic Routing

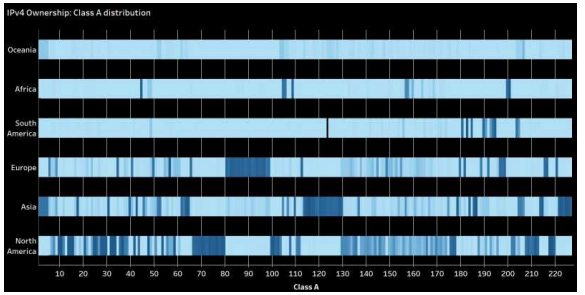
https://en.wikipedia.org/w/index.php?title=Open_Source_Geospatial_Foundation

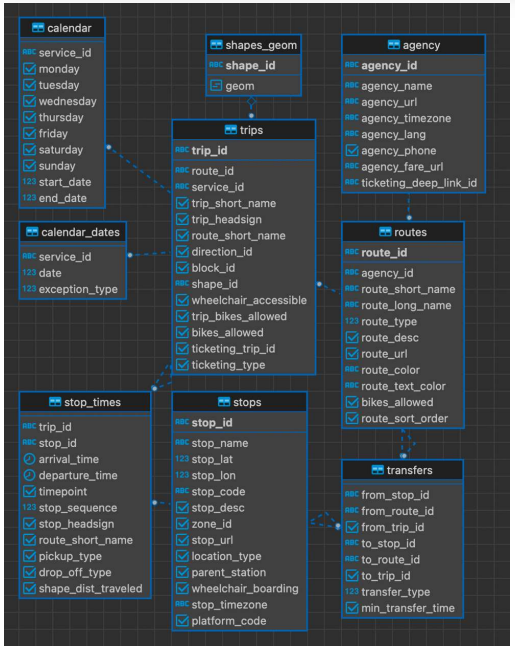
macOS Tools

- Terminal
- Sublime or another text editor
- QGIS
- Tableau, Excel and/or Jupyter Notebook for data viz
- Dbeaver
- Homebrew to install command-line tools (Python, GDAL, pgRouting, jq)









macOS vs Windows

- No exclusive software on macOS (except Homebrew).
- Homebrew on macOS has fewer steps to install latest DBs and GIS tools than Ubuntu or Ubuntu on Windows.
- ArcGIS Pro's Desktop app runs on Windows; Browser version for everyone else.
- QGIS has newer libraries with more features on Windows 10 (GDAL 3.5+, etc.)
- 6+ disk drives aren't uncommon for Windows machines.

Thank you

tech.marksblogg.com