Ah Mapbox, here we go again! How to make Mapbox work with large datasets

A detailed explanation

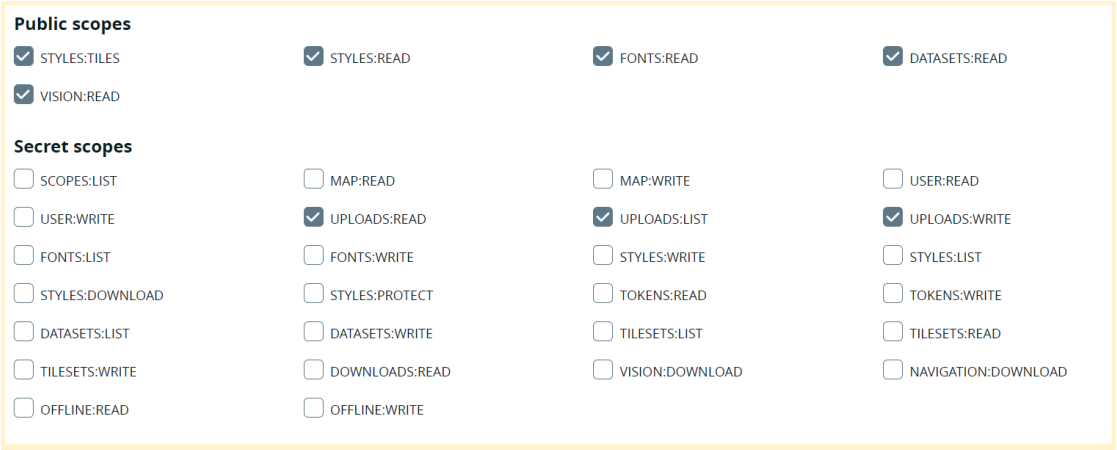
ANA RUIZ - DATAHARVEST 2024

(*) Before starting:

1. **Download the files that we are working with today from this link.**
2. **Mapbox credentials (in case you do not have a Mapbox account):**
   - Username: anadelarue98
   - Api key (password that will allow us to upload the information later): sk.eyJ1IjoiYW5hZGVsYXJ1ZTk4IiwiYSI6ImNsd25uNzZ4bDAycnoyaW40dHNlcXJqcGwifQ.FcCzc_HUlcdsvjJRfD2otQ

Steps:

1. **Create a Mapbox account:** this would be the first logic step. Nonetheless, if you do not have one, for today's session you can use the credentials above. If you have a Mapbox account, you can create your own token.
   To do this, you need to go to Account > Create Token. Once there, you have to create a token with scope uploads.



2. **Open the terminal and install node.js and npm:**
   (*) Node.js is a runtime environment that allows JavaScript to run outside of web browsers, enabling developers to build diverse applications like web servers, command-line tools, and IoT devices.
   npm, the Node Package Manager, is an essential tool accompanying Node.js, facilitating the discovery, installation, and management of reusable JavaScript code packages, streamlining development workflows.

   _____

   To install node.js and npm:

```
$ sudo apt install curl
```

```
$ curl -s https://deb.nodesource.com/setup_20.x | sudo bash
$ sudo apt install nodejs
```

3. ***Install ogr2ogr to transform files into geojson files (this step is optional):***
   In the project we applied the steps that we are following today, we worked with some geographic xml files, called .gml. To upload them to Mapbox, we first had to transform them into geojson files. For that, we used the ogr2ogr tool.
   We will be using geojson files today and this step is not needed, but you may find the code useful.
   The installation (Linux):

```
$ sudo apt update
$ sudo apt install gdal-bin
$ ogr2ogr - -version
```

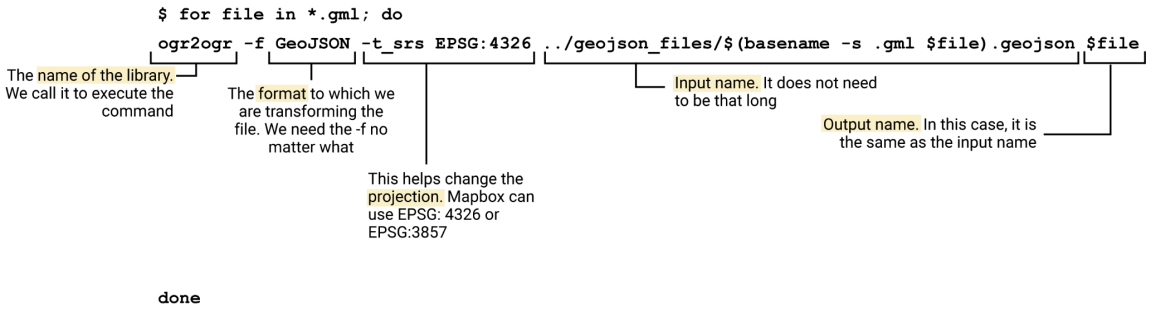
The installation (Mac):

```
$ brew install gdal
$ pip download GDAL
$ tar -xpzf GDAL-<version of GDAL>.tar.gz
$ cd GDAL-<version of GDAL>
$ python setup.py build_ext --gdal-config
/usr/local/Cellar/gdal/<version of GDAL>/bin/gdal-config
$ python setup.py build
$ python setup.py install
```

The command:

```
$ for file in *.gml; do
      ogr2ogr -f GeoJSON -t_srs EPSG:4326
../geojson_files/$(basename -s .gml $file).geojson $file
done
```

The code explained:



4. **Install [mapshaper]**

Mapshaper is a tool that helps people edit and simplify geographic data, like maps, without needing to know complex programming.
Link to documentation

The installation (Linux y Mac):

```
$ (sudo) npm install -g mapshaper
$ mapshapper — version
```

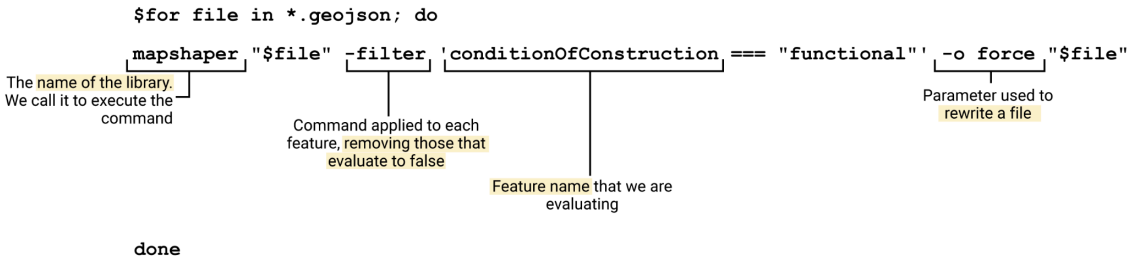
(*) The command "sudo" would not be necessary in Mac.

5. **Keep only features equal to "functional":** This would be our first step, since this column will not be part of the file uploaded to Mapbox. We only keep the features equal to "functional", because the files also include if the building is "ruinous" or "empty". For this we use -filter.

   The command:

   ```
   $for file in *.geojson; do
       mapshaper "$file" -filter 'conditionOfConstruction === "functional"' -o force "$file"
   done
   ```

   The code explained:

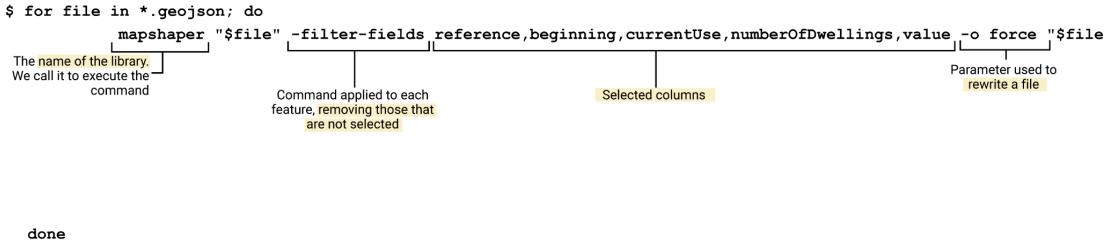
   

6. **Keep only the columns that we need:** In this case, there are many columns with redundant or unnecessary information that we do not want to keep. To select columns with mapshaper, we use -filter-fields.

   The command:

   ```
   $ for file in *.geojson; do
         mapshaper "$file" -filter-fields
         reference,beginning,currentUse,numberOfDwellings,value -o
         force "$file"
   done
   ```

   The code explained:

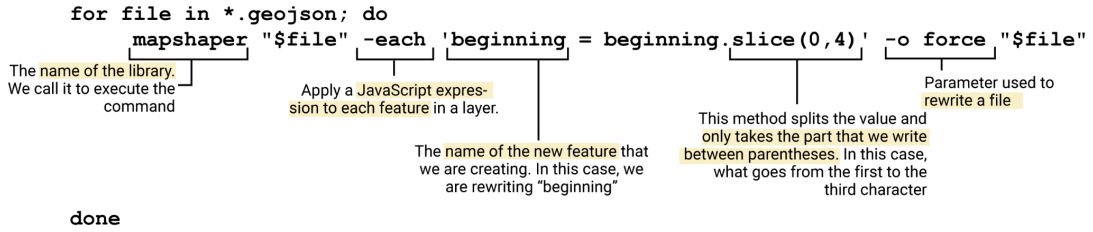


7. **Transform the feature "beginning" to year format:** this feature contains the year, month, day, hour, minute and second in which the house was created, but only the first piece of information varies. Therefore, this is the only part that we need. To keep only the first four digits of the feature we will use the command -each, along with the JS method split.

   The command:

```
$ for file in *.geojson; do
mapshaper "$file" -each 'beginning = +beginning.slice(0, 4)'
-o force "$file"
done
```

   The code explained:

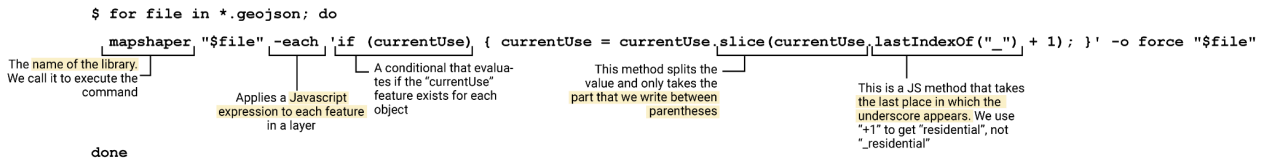


8. **Delete other residual information:** in this case, er have a feature called "currentUse" that contains information like this "1_residential" or "4_3_retail". If we only want to keep the last part, we need to tell mapshaper to only keep the text that goes after the underscore.

   The command:

```
$ for file in *.geojson; do
  mapshaper "$file" -each 'if (currentUse) { currentUse =
currentUse.slice(currentUse.lastIndexOf("_") + 1); }' -o force
"$file"
done
```

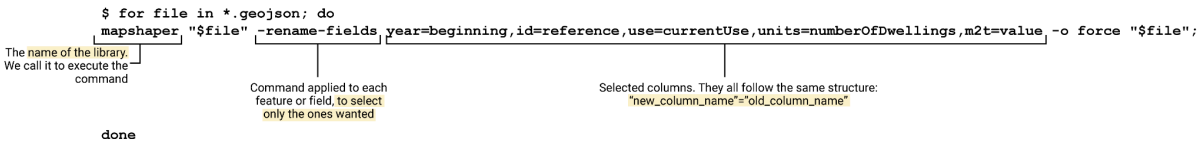
   The code explained:



9. **Rename columns:** give the columns name that take less space. We will do it with -rename-fields.
   The command:

```
$ for file in *.geojson; do
 mapshaper "$file" -rename-fields
year=beginning,id=reference,use=currentUse,units=numberOfDwell
ings,m2t=value -o force "$file";
done
```

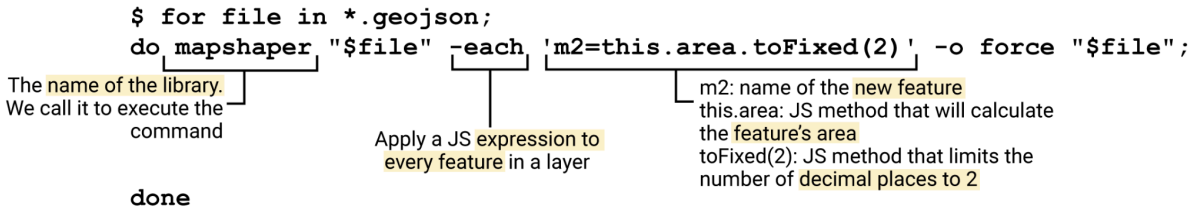
   The code explained:



10. **Create a feature with the surface area:** the area we have right now is the one of the whole building. That means that, if we have a building block with five floors and 100 m2 each, the feature "value" will display 500m2, not 100m2. We want this last value.
    The command:

```
$ for file in *.geojson;
do mapshaper "$file" -each 'm2=this.area.toFixed(2)' -o force
"$file";
done
```

    The code explained:



11. **Reduce the coordinates' decimal numbers:** the extra decimal numbers take extra space that we do not need. Usually, a bigger number represents more precision when the features are displayed. Anyway, Mapbox can represent small features with great precision with only six decimal places.

    The installation:

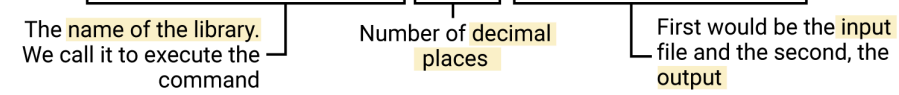
```
$ npm install [-g] geojson-precision
```

    The command:

```
$ for file in *.geojson;
do geojson-precision -p 6 "$file" "$file";
echo “$file”;
done
```

The code explained:

```
$ for file in *.geojson; do
     geojson-precision -p 6 "$file" "$file";
```




```
done
```

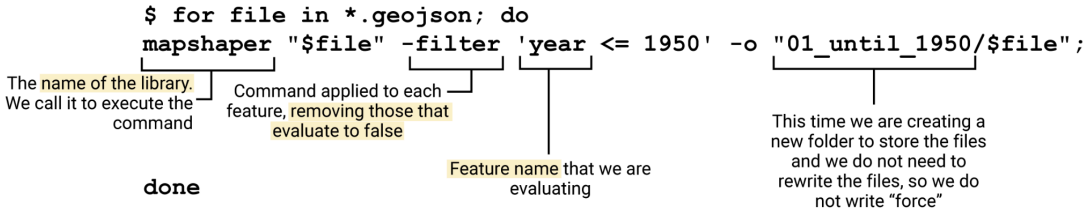
12. **Divide all information in different layers:** Mapbox imposes limits on the size of vector tiles to ensure performance and stability. The maximum size for a vector tile is 500 KB uncompressed. If a tile exceeds this size, Mapbox will drop some features, including polygons, to reduce it. This limit applies to individual vector tiles.
    The most helpful tip to exceed this limit is to create different layers, since the limitation will be applied to each one of them individually.
    For this, we will use again, the command -filter from the library mapshaper.

    The command:

    ```
    $ for file in *.geojson;
    do mapshaper "$file" -filter 'year <= 1950' -o
    "01_until_1950/$file";
    done
    ```

    The code explained:

    ```
    $ for file in *.geojson; do
    mapshaper "$file" -filter 'year <= 1950' -o "01_until_1950/$file";
    ```


    ```
    done
    ```

13. **Transform data to mbtiles format with Tippecanoe:**
    Tippecanoe is a tool that helps people turn large sets of geographic data into interactive maps that can be easily viewed and explored online. It was created by Mapbox itself and helps you control zoom levels, polygon dropping and more.
    The command:

#To install the library in Linux:

```
$ git clone https://github.com/mapbox/tippecanoe.git
$ cd tippecanoe
$ make -j
$ make install
```

#To install the library in Mac:

```
$ brew install tippecanoe
```

#To execute the command:

```
$ tippecanoe -zg -Z5 -o until50.mbtiles -l until50
--drop-densest-as-needed --no-tiny-polygon-reduction --hilbert
--reorder --coalesce *.geojson
```

The code explained:

- **tippecanoe:** The name of the library. We call it to execute the command.
- **-zg:** maximum zoom level.
- **-Z5:** minimum zoom level. In this case, 5.
- **-o until50.mbtiles:** this is how we specify the output name. We need the "-o" to make sure it is the output.
- **-l until50:** the layer name.
- **--drop-densest-as-needed:** at higher zoom levels, it will drop the polygons from the densest areas.
- **--no-tiny-polygon-reduction:** at higher zoom levels with big polygon concentrations, it will not simplify the polygon geometry.
- **--hilbert:** Put features in Hilbert Curve order instead of the usual Z-Order. This improves the odds that spatially adjacent features will be sequentially adjacent, and should improve density calculations and spatial coalescing. It should be the default eventually.
- **--reorder:** Reorder features to put ones with the same attributes in sequence (instead of ones that are approximately spatially adjacent), to try to get them to coalesce.
- **--coalesce:** Coalesce consecutive features that have the same attributes. This can be useful if you have lots of small polygons with identical attributes and you would like to merge them together.

14. Upload to Mapbox with MapboxCLI

MapboxCLI is a command line interface to Mapbox Web Services. There are other ways of interacting with Mapbox through the command line, such as Mapbox Tiling Service or the Uploads API.

(*)Note: the tool that we are going to use is deprecated for the python and pip versions installed. Therefore, we will have to make some adjustments to the library itself before working on it.

The commands:

#For the installation in Linux:

```
$ pip install mapboxcli
```

#For the installation in Mac:

```
$ brew install mapbox/cli/mapbox
```

#To fix our little problem:

```
$ cd /usr/local/lib/python3.10/dist-packages/mapbox
```

```
$ sudo nano utils.py
```

```
$ add .abc a collections
```

#Define our token as a global variable, so Mapbox knows we have access and allows us to upload the file:

```
$ export MAPBOX_ACCESS_TOKEN=ACCESS_TOKEN
```

In our case it would be:

```
$ export
MAPBOX_ACCESS_TOKEN=sk.eyJ1IjoiYW5hZGVsYXJ1ZTk4IiwiYSI6ImNsd25
uNzZ4bDAycnoyaW40dHNlcXJqcGwifQ.FcCzc_HUlcdsvjJRfD2otQ
```

#Upload files:

```
$ mapbox upload {username}.{desired_layer_name}
{file_name}.mbtiles
```

In our case it would be:

```
$ mapbox upload anadelarue98.until50 until50.mbtiles
```