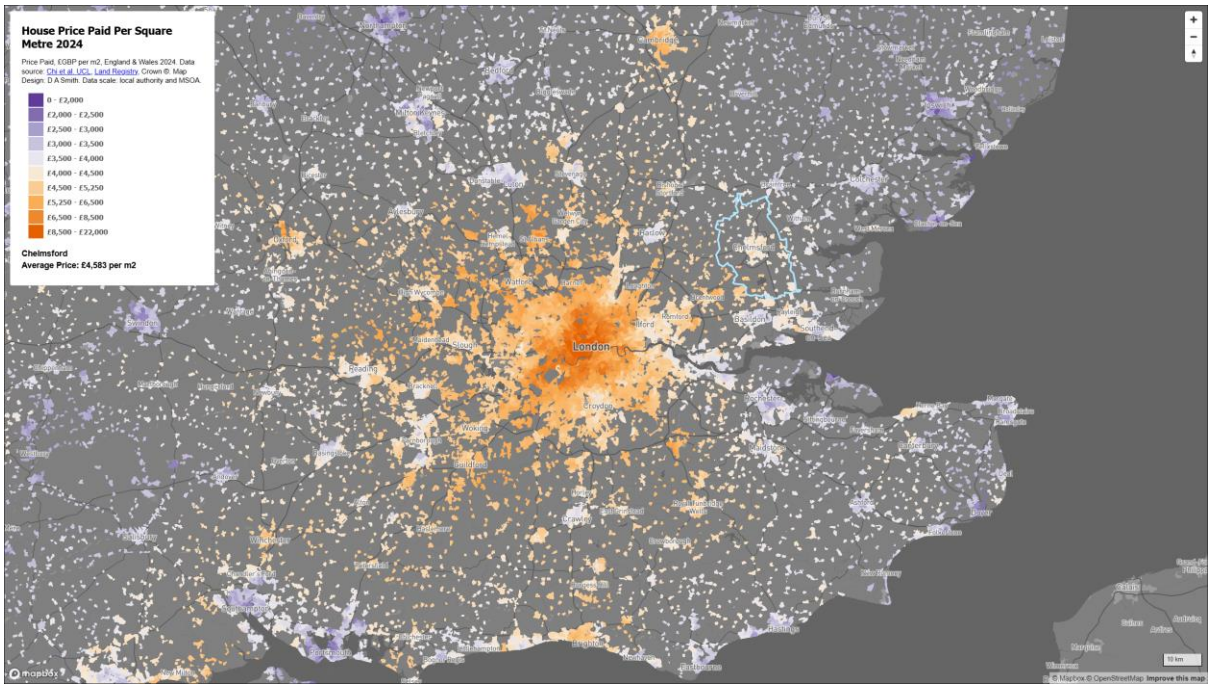
CASA0029: Urban Data Visualisation
Advanced Interactive Mapping Practical

This practical continues to develop your interactive mapping skills using Mapbox Studio and the JavaScript library Mapbox GL. We will look at adding data at different levels of detail; creating 2.5D buildings in Mapbox Studio; and adding a 2.5D density layer in Mapbox GL. The files used in this practical are on Moodle named- 'Practical 4 Data & Examples 2025-26'. Unzip this file, and you will see the data files and the HTML examples.

Choropleth Mapping Example: House Prices Per Square Metre

First of all, we are going to revisit the House Price Per Square Metre choropleth map example from a couple of weeks ago. This example demonstrates techniques to ensure data layers are visible across all zoom levels. There is a data limit for each vector tile, and if you use detailed polygon geometry then the data can often have limited visibility at low zoom levels. This can be avoided by using lower detail larger zones for low zoom levels, and also by using simplified urban boundaries.

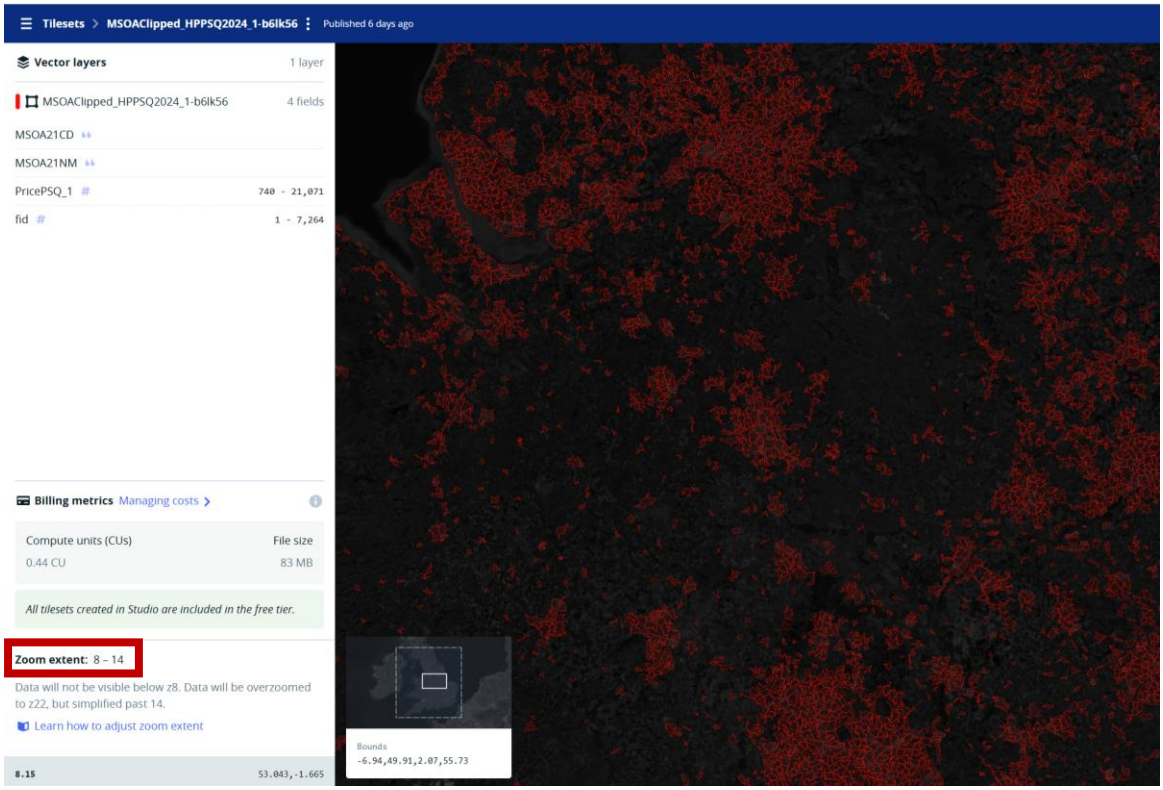
You will see that the zonal data has been clipped to urban boundaries. This is a useful technique for urban data visualisation, so that the user can see how the variable being visualised changes between cities, towns and rural areas. The urban outlines data comes from Ordnance Survey data on Digimap, using the Meridian 2 dataset. The OS also has a more detailed "OS Open Built-up Areas" dataset, which is useful for high zoom levels, but is too detailed here for lower zoom levels. The urban outlines are used to "Clip" the local authority and MSOA data, using the Clip tool in QGIS.



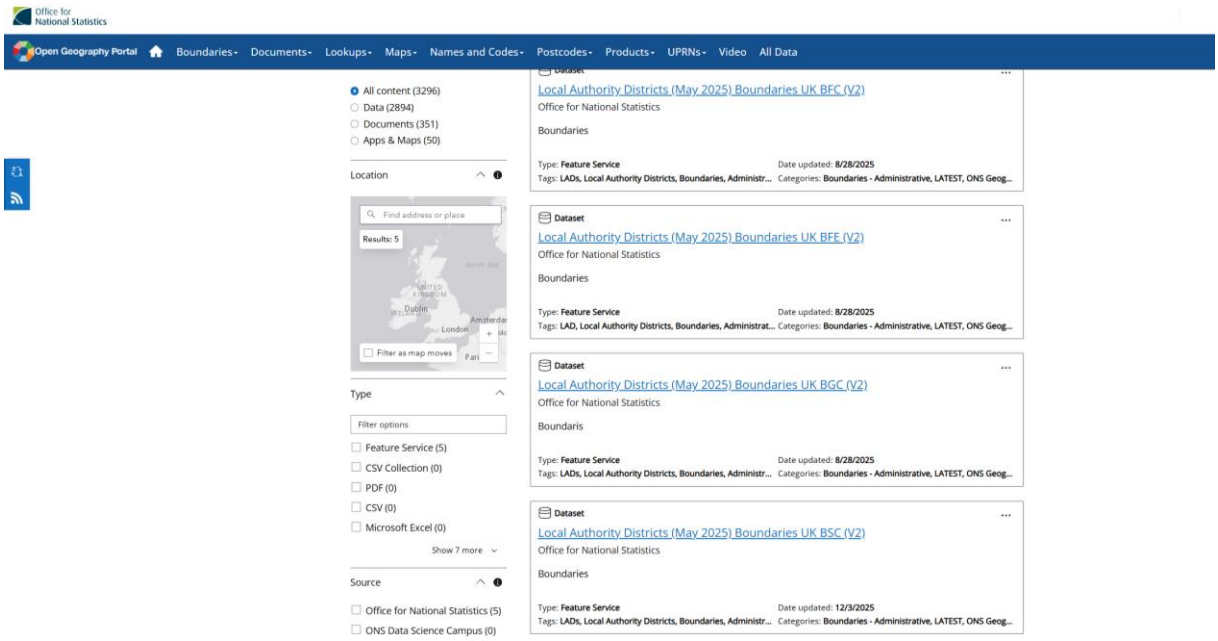
The House Price per Square Metre data comes from combining the Land Registry Price Paid data with the Energy Performance Certificate floorspace data, achieved in a research project at CASA UCL- https://data.london.gov.uk/dataset/house-price-per-square-metre-in-england-and-wales-epo9w/

Zoom Levels and Level of Detail for Vector Tiles

A challenge when mapping more complicated geometry using vector tiles is that there is a maximum amount of data that can be included in each vector tile. This means it is more difficult to visualise detailed vector data as you zoom out, and Mapbox can limit the visibility of layers at low zoom levels. Download the 'Practical 4 Data & Examples' from Moodle, and upload the two House Price layers as Tilesets in Mapbox. You will see that the MSOA data used in the House Price Map has a minimum zoom level of 8, which means that the layer is not visible at zoom levels below this-



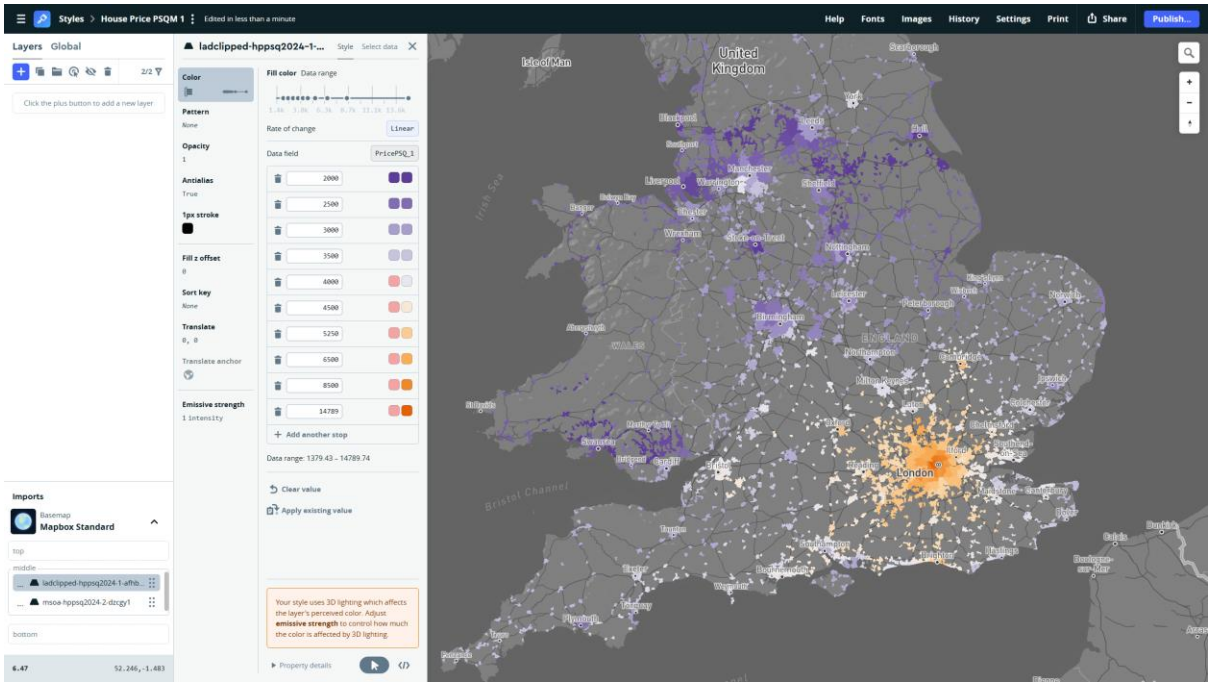
The less detailed local authority layer data however has a Zoom Extent of 0-11, which means it will be visible across all zoom levels (it will still be visible past the max zoom level), allowing all of the UK to be viewed in one map. The Local Authority layer uses low detail zones from ONS Geoportal-



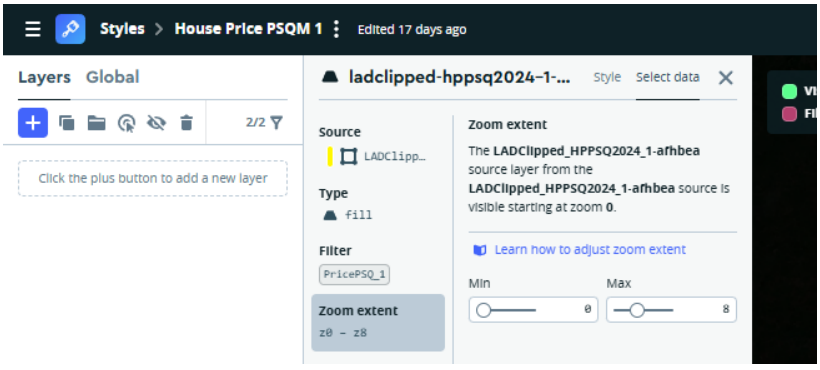
The boundary layers on the ONS Geoportal are marked BFC (full level of detail, clipped to coastline); BGC (generalised, clipped to coastline) and BSC (super-generalised, clipped to coastline). Here we have used the super-generalised option to get the most simplified vector data. This is then visible on Mapbox across all zoom levels.

Designing the Map in Mapbox Studio

The map was designed in Mapbox Studio with two custom layers: the Local Authority layer and the MSOA layer. These were styled across data range using the "PricePSQ_1" (price per square metre) column. A diverging ColorBrewer colour scheme has been used (with the colour codes copied and pasted from QGIS) to emphasise high and low price contrasts across England and Wales. Note you can copy and paste styles between layers so you can transfer styles between the local authority and MSOA layers. This means you do not have to input the colour scheme twice.



We want the Local Authority and MSOA layers to swap over as the user zooms in. This is achieved using the maximum and minimum zoom settings on the "Select data" tab. The local authority layer has a max zoom of 8, and the MSOA layer has a min zoom of 8, so the two layers are switched at zoom level 8-



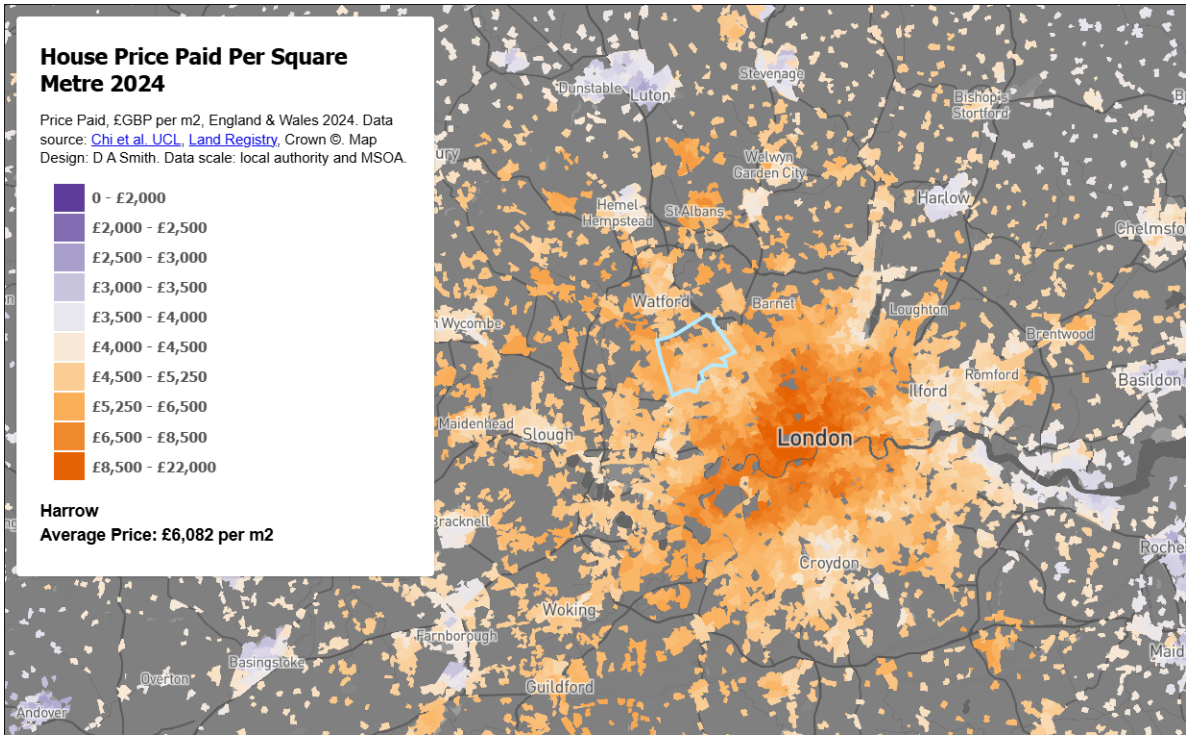
You can also see in the example that there are city labels overlaid on top of the data. This is achieved by dragging the custom layers to the 'middle' section (bottom-left) of the layers panel. This means

that Mapbox place labels will appear above the layer, and so the user is provided with a hierarchy of place labels without us having to add labels manually.

Creating an Interactive Hover Effect in Mapbox.GL JS

Have a look at the HTML code for "Mapbox_example5_HousePriceMap.html". You will see that there is HTML code for the legend, with a title, data attribution and breakdown of the prices for the different colours on the map. Then there is the standard code we have been using to add the Mapbox style of the house price data.

In the "map.on('load', function()" section, there is code that creates interactive functionality where the user can see the average price of the zone their mouse cursor is hovering over. The name and price is shown in the legend, and a blue outline shows the user the zone that is highlighted.

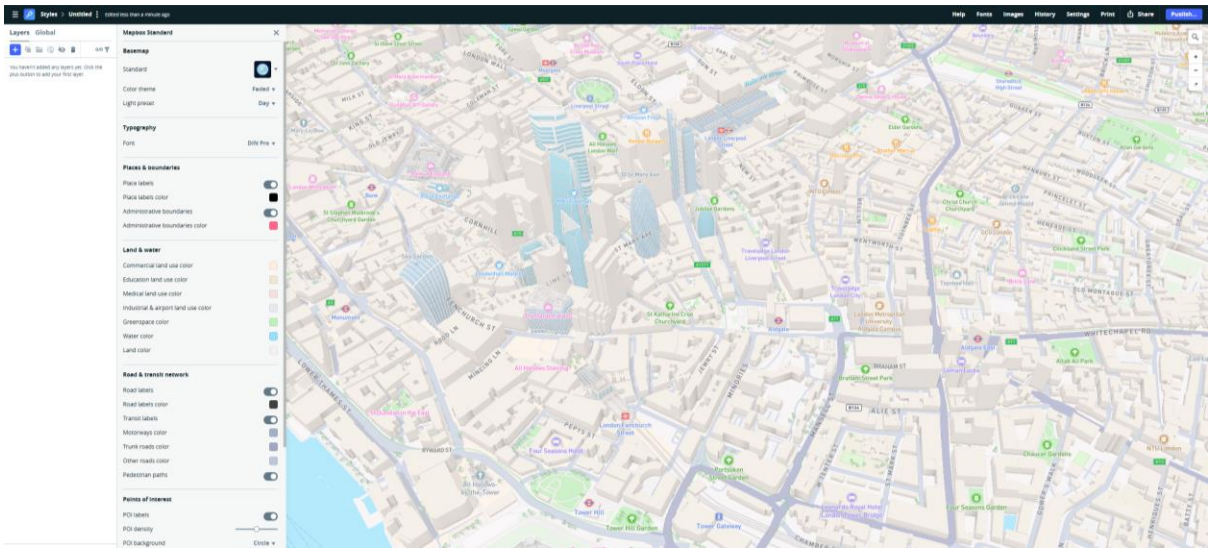


This is achieved in the Mapbox.gl JS code by adding Tilesets for the Local Authority and MSOA layers. This is done twice, once as an invisible fill layer, which is used to identify which zone the user is hovering over, and a second time as a line layer which shows the user the zone the highlighted layer. The line layer has an initial filter removing all features.

The highlight functionality is based on an event listener, an 'on mousemove' function. This returns a point of where the user's mouse pointer is. A 'queryrenderedfeatures' function asks whether there is a local authority or MSOA feature below this point. If there is a feature, then this is used to set a new filter for the highlight layer (displaying the outline of the selected feature) and also change the HTML for the name and average price for this feature and updates the highlighted zone using a filter. This piece of code is based directly on this Mapbox.gl examples-
https://docs.mapbox.com/mapbox-gl-js/example/hover-styles/
https://docs.mapbox.com/help/tutorials/choropleth-studio-gl-pt-1/?step=0

Adding 2.5D Buildings in Mapbox Studio

Mapbox has 3D buildings included in its standard basemap in Mapbox Studio. The 3D buildings include 2.5D buildings derived from OpenStreetMap, and an additional layer of more detailed landmark buildings for many global cities-



While the 3D buildings are aesthetically impressive, there are quite a few limitations with this default buildings layer for data visualisation. Height data is not available for all cities across the globe. Additionally, OpenStreetMap has limited building attributes, and we are unable to add our own building attributes using this method. And lastly, if you zoom out you will notice that at low zoom levels the buildings are no longer visible. We can upload our own buildings layers to try and improve on the default layer.

Adding A Custom Buildings Layer to Mapbox Studio

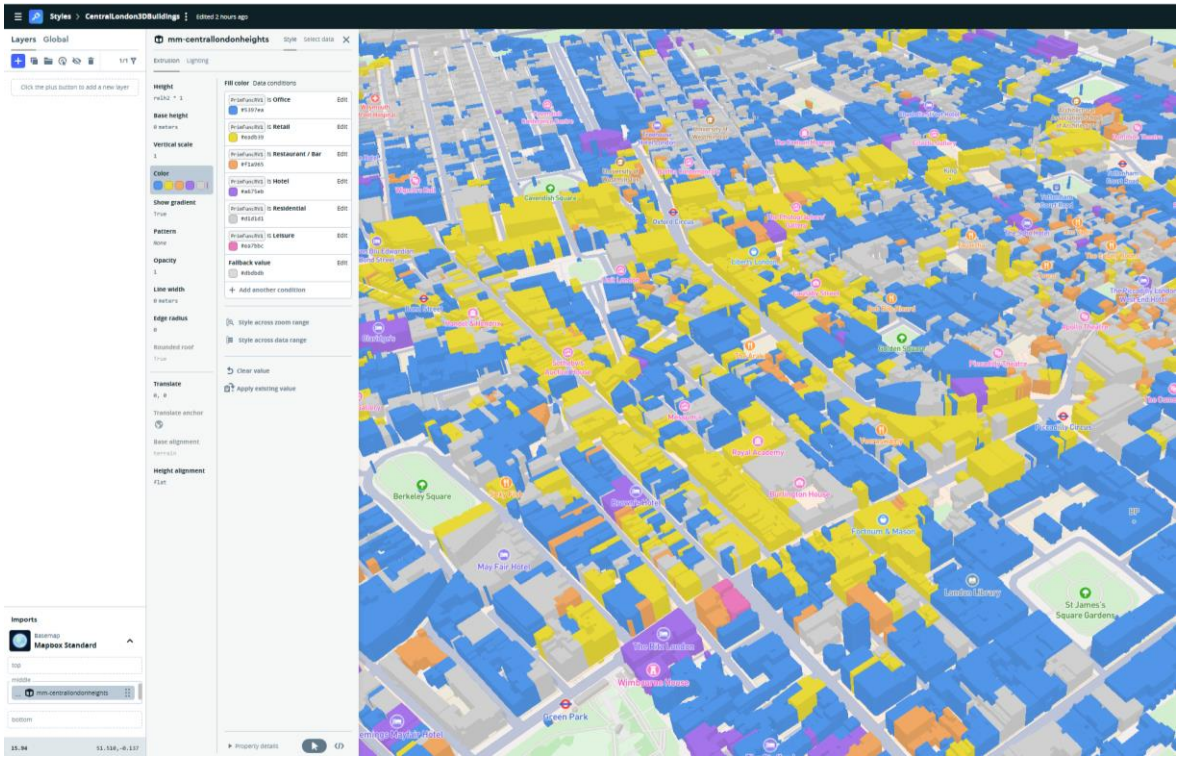
A typical approach for creating basic 2.5D urban models is to use LIDAR data. LIDAR data for UK cities is available (produced by the Environment Agency) from Digimap-
https://digimap.edina.ac.uk/lidar

EDINA have also created a layer joining LIDAR data with Mastermap building outlines which is available via Digimap through the Ordnance Survey download interface. Note that this layer is not open data, so should not be used for public facing websites. An open data version could be created using alternative building outline data, such as Ordnance Survey Open Map layers.
https://www.ordnancesurvey.co.uk/products/os-mastermap-building-height-attribute

An example of the building heights dataset is available on Moodle named-LondonBuildingHeights.mbtiles. The 'mbtiles' file is the Mapbox standard for vector tiles. It is a type of SQL Lite database used to store Mapbox tile layers. This file was created by importing the London Building Heights data into a program called Tippecanoe. Tippecanoe is used to generalise vector data, and allow Mapbox layers to be visible across a wider range of zoom levels. The documentation of Tippecanoe is here (unfortunately this software is Mac only)-
https://github.com/mapbox/tippecanoe

There is also a more advanced Mapbox Tiling Service that can achieve a similar outcome-
https://docs.mapbox.com/mapbox-tiling-service/guides/

Mbtiles files can be directly uploaded to Mapbox as TileLayers. Upload the LondonBuildingHeights.mbtiles file to a new TileLayer, and add the file to a Mapbox Studio style. When you add the London Building Heights file, you will notice it overcomes several of the limitations of the default Mapbox layer. It is visible across a wider range of zoom levels, height information is available for all building polygons, and some land use information has been included as an example of an additional building attribute-



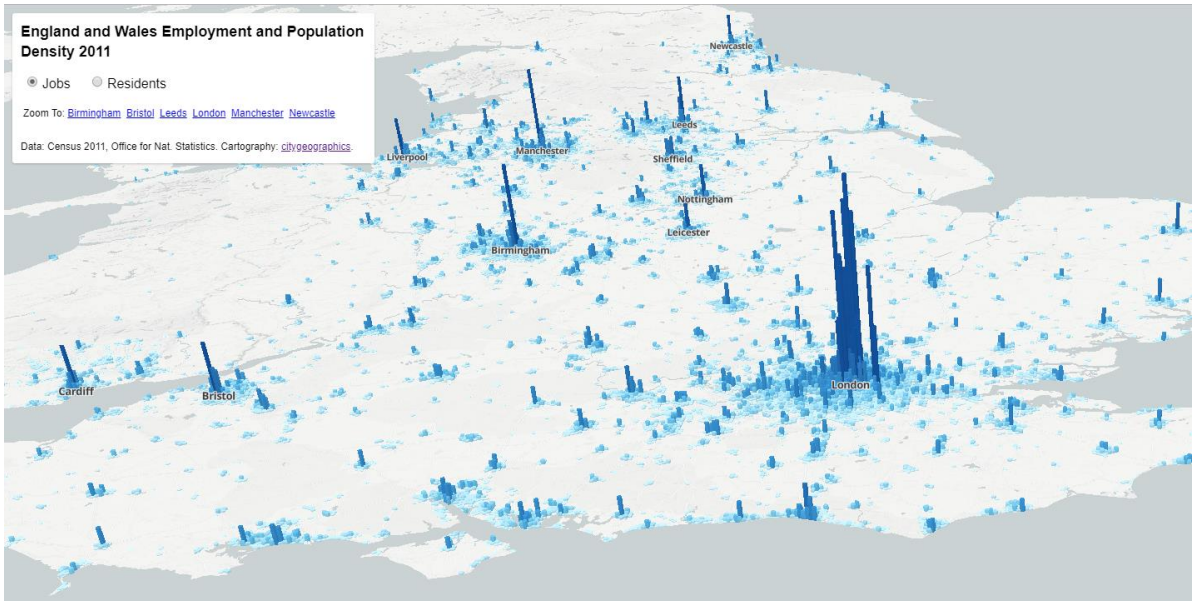
You could potentially visualise millions of buildings using this method (see for example https://netherlands.parallel.co.uk/). It can be challenging to geolocate data in the UK at building level (compared to geolocating by census zones or postcodes). The land use data was created as a quick example by joining data from the Valuation Office at postcode level. This is OK for a general idea of urban functions, but leads to errors for individual buildings. Generating improved individual building level data would be an interesting topic for projects (this is similar to a research project at the Bartlett called https://colouring.london/).

Another interesting issue to explore adding landmark buildings to improve the 2.5D model. Mapbox does have capabilities to add 3D models in combination with the library three.js-
https://docs.mapbox.com/mapbox-gl-js/example/add-3d-model/
https://docs.mapbox.com/mapbox-gl-js/example/clip-layer-building/

Styling a 2.5D Layer in Mapbox GL

As well as creating extruded layers in Mapbox Studio, this can also be achieved via Mapbox GL on the client side in JavaScript. This example uses a 1km^2 hexagon layer of the density of population and jobs in England and Wales. More details on how the hexagon density layer is created are here-

https://citygeographics.org/luminocity/luminocity3d-spatial-analysis/



The hexagon density layer is added as a 'fill-extrusion' layer, with the height set in the paint properties. In fact, there are two fill-extrusion layers added, one for residential population and another for jobs. The radio button at the top right switches between the layers via an opacity control.

There is also a feature allowing the user to zoom to different cities. This is achieved using the map.flyTo() method.

Mapbox_example7_3DDensity.html

```
<!DOCTYPE html>
<html>
<head>
    <meta charset='utf-8' />
    <title>England and Wales Employment and Population Density 3D Map</title>
    <meta name='viewport' content='initial-scale=1,maximum-scale=1,user-scalable=no' />
    <script src="https://api.tiles.mapbox.com/mapbox-gl-js/v3.3.0/mapbox-
gl.js"></script>
    <link href="https://api.tiles.mapbox.com/mapbox-gl-js/v3.3.0/mapbox-gl.css"
rel="stylesheet" />
    <style>
…
    </style>
</head>
<body>


<div id='map'></div>

<div class='map-overlay'>
```

```
    <div class='map-overlay-inner'>
        <h2>England and Wales Employment and Population Density 2011</h2>
        <table><tr><td>
        <input type="radio" name="layers" id="layer1" value="Employment"
checked><label>Jobs    </label>
        <input type="radio" name="layers" id="layer2" value="Residents"><label>Residents
</label>
         </td>
         </tr>
         <tr><td><p class="cityfly">Zoom To: <a href="#" class="citylink"
id="birm">Birmingham</a>  <a href="#" class="citylink" id="bris">Bristol</a>
 <a href="#" class="citylink" id="leed">Leeds</a>  <a href="#"
class="citylink" id="lond">London</a>  <a href="#" class="citylink"
id="manc">Manchester</a>  <a href="#" class="citylink" id="newc">Newcastle</a></p>
         </td></tr>
         </table>
         <p class="credit">Data: Census 2011, Office for Nat. Statistics. Cartography:
<a href="http://citygeographics.org">citygeographics</a>.</p>
</div>
</div>

    <script>
    mapboxgl.accessToken =
'pk.eyJ1IjoiZHVuY2FuMjAwMSIsImEiOiIyRDhtOE1nIn0.OA5QmCprkPbOoxZog8HIow'; // Put your
Mapbox Public Access token here

    // Load a new map in the 'map' HTML div
    var map = new mapboxgl.Map({
    container: 'map', // container id
    style: 'mapbox://styles/duncan2001/cjckdm9hn24qo2qmswnj13555', // map background
layer location
    center: [0, 51.5], // starting position [lng, lat]
    zoom: 8, // starting zoom
    pitch: 50 // tilt of the viewpoint in degrees
    });


    map.on('load', function() {
      // Set global light properties which influence 3D layer shadows
      map.setLight({color: "#fff", intensity: 0.15, position: [1.15, 210, 30]});
      // Add standard navigation control
      map.addControl(new mapboxgl.NavigationControl());

    // Load the 3D employment hexagon layer as a fill-extrusion type
      map.addLayer({
        id: 'EngWal_Hex_Emp',
        type: 'fill-extrusion',
        source: {
          type: 'vector',
          url: 'mapbox://duncan2001.92wzi5h7' // Your Mapbox tileset Map ID
        },
        'source-layer': 'EngWal_Hex_ResEmp_20012011b-7qjjyc', // name of tileset
        paint: {
            'fill-extrusion-color': {
                property: 'Emp2011',
                type: 'exponential',
                stops: [
                    [0, '#d6f5ff'],
                    [3000, '#69cbf5'],
                    [5000, '#2e96dc'],
                    [30000, '#00479e']]
            },
            'fill-extrusion-height': ['/', ['number', ['get', 'Emp2011'], 2],
2],  //expression that controls the extrusion height based on employment attribute
Emp2011
```

```
            'fill-extrusion-opacity': 0.95,
            'fill-extrusion-opacity-transition': {  //Opacity transition adds a delay
when changing the opacity for a smooth layer change effect
                 duration: 1000,
                 delay: 0
             }
            }
      });

    // Load the second 3D population hexagon layer. This layer has its opacity set to
zero.
      map.addLayer({
        id: 'EngWal_Hex_Res',
        type: 'fill-extrusion',
        source: {
          type: 'vector',
          url: 'mapbox://duncan2001.92wzi5h7' // Your Mapbox tileset Map ID
        },
        'source-layer': 'EngWal_Hex_ResEmp_20012011b-7qjjyc', // name of tileset
        paint: {
            'fill-extrusion-color': {
                property: 'Res2011',
                type: 'exponential',
                stops: [
                    [0, '#fff2f2'],
                    [5000, '#f59c8e'],
                    [10000, '#e54545'],
                    [30000, '#730b0b']]
            },
            'fill-extrusion-height': ['/', ['number', ['get', 'Res2011'], 2], 2],
            'fill-extrusion-opacity': 0,  //Opacity set to zero
            'fill-extrusion-opacity-transition': {
                 duration: 1000,
                 delay: 0
             }
            }
      });


// Add the label layer
    map.addLayer({
        'id': 'labels',
        'type': 'symbol',
        source: {
          type: 'vector',
          url: 'mapbox://duncan2001.b6rqk9s2' // Your Mapbox tileset Map ID
        },
        'source-layer': 'LabelCities2-6qmjf4', // name of tilesets
        'layout': {
            'text-field': '{Name2}',
            'text-font': ['Open Sans Bold', 'Arial Unicode MS Bold'],
            'text-size': 14
        },
        'paint': {
            'text-color': 'rgba(0,0,0,0.8)',
            'text-halo-color': '#fff',
            'text-halo-width': 1
        }
    });

//Event listener for layer switch
document.getElementById("layer1").addEventListener("click", function(){
map.setPaintProperty('EngWal_Hex_Emp','fill-extrusion-opacity',0.95);
map.setPaintProperty('EngWal_Hex_Res','fill-extrusion-opacity',0);
});
```

```
document.getElementById("layer2").addEventListener("click", function(){
map.setPaintProperty('EngWal_Hex_Emp','fill-extrusion-opacity',0);
map.setPaintProperty('EngWal_Hex_Res','fill-extrusion-opacity',0.95);
});


//Event listener for the zoom to buttons created using a for loop and switch case
statement to set lat and long
var x = document.getElementsByClassName('citylink');
var i;
        for (i = 0; i < x.length; i++) {
            x[i].addEventListener('click', function(e) {

                var lat,long;

                switch(e.target.id) {
                    case "birm": long=-1.8904; lat=52.4862; break;
                    case "bris": long=-2.5879; lat=51.4545; break;
                    case "leed": long=-1.5491; lat=53.8008; break;
                    case "lond": long=-0.1278; lat=51.5074; break;
                    case "manc": long=-2.2426; lat=53.4808; break;
                    case "newc": long=-1.6178; lat=54.9783; break;
                }

                map.flyTo({
                    center: [long,lat],
                    zoom: 9,
                    speed: 0.3,
                    pitch: 50
                    });
            });
        }

});

</script>

</body>
</html>
```

Further Mapbox Examples

We have covered some more advanced mapping examples, with data layers at different levels of detail, interactive query functionality, and 2.5D buildings. There are many further examples of Mapbox GL functionality online to explore, on the main example pages and through various blogs that showcase recent developments in interactive mapping-

https://docs.mapbox.com/mapbox-gl-js/example/

https://docs.mapbox.com/mapbox-gl-js/api

https://blog.mapbox.com/