Automated generation

of

Equal Area Cartograms (hexmaps) at any scale

Bruce Mitchell MSc
ONS GeoCentre
November 2022

Table of Contents

Automated generation of Equal Area Cartograms (hexmaps)

WHY MIGHT YOU BE INTERESTED?

- Do you need to display data on a map?
- Is the geography you need to map made up of sub-units of widely differing sizes?
- Are the most interesting data to be found in the smaller areas?
- Alternatively, are your data located at point that are by turns tightly concentrated and widely scattered?

If the above apply, then this is definitely for you.

We have created a tool written in Python to create equal area cartograms (EACs) based on hexagonal grids. The purpose is to get around the visual dominance of large areas on maps, and the tendency for smaller areas to be overlooked. This document explains the nature of the problem and considers alternative solutions before turning to hexmaps.

The ONS approach has been successfully tested on a wide range of geographies with an extreme range of scales.

We explain how to set up the script and your data to run on any PC running 64-bit Windows 7 or 10.

We then go through some examples to demonstrate how the methodology works.

We are separately developing an R-based method for automating the production of square-based (waffle chart) EACs.

Acknowledgements

I conceived both projects in 2016-2017 and worked out the hexmap methodology with George Tzelepis. George wrote the original code in ArcPy for ESRI's ArcGIS. He subsequently ported the product over to Python, but development was then critically delayed by IT issues within ONS. George left ONS and ceased active development in May 2018. Since then, a way has been found around the IT issues and this has allowed me to test and further develop the code.

So, in the first rank, I must thank George Tzelepis for doing the heavy lifting of working out how to convert my original ideas into code – not once, but twice – and working out the geometry. Further thanks are due to Andy Harfoot of Geodata Institute (University of Southampton) has also assisted with the crystallising some of the ideas and correcting some of the code. I'd also like to thank ONS Geography managers Andy Tait, Nick O'Rourke, Chris Gale and Alistair Calder for their support and patience over the project's long gestation

Introduction

Thematic mapping draws attention to statistical difference across space. Statistical values of interest are often attached to sub-areas of the overall geography, and this may reveal significant spatial variations. However, administrative divisions commonly encompass a range of unit sizes, from small urban districts to large rural expanses.

The tyranny of large areas

When seen together on a map, larger units tend to dominate and the smaller to recede. The eye is drawn to large areas. Think of Canada, Australia and Russia, where huge areas, often relatively sparse in socio-economic statistics, acquire visual dominance at the expense of the smaller and often more populated areas. Those familiar with the geostatistics behind any given map may be able to apply a mental filter to compensate, but the fact remains that the areas where the chief interest of your story resides may be small and hard to see.

Here in the UK, it is a similar story. Local Authority Districts (LADs) in the UK range in size from the 290 hectares of the City of London to the 2.6 million hectares of the Highlands of Scotland – but the two are administratively equivalent. On a national map, the City of London is completely invisible.

British statistical geographies (Census output areas) are just as vulnerable to the issue as the administrative geographies. They are based on population thresholds, so are made up of units (instances) which are small where the population is dense, and large where it is spread thinly. At the middle level [MSOA (EW), DataZones (Sc) and Small Areas (NI)], they range from just over 1 hectare (ha) to 180,000 ha,

Impact upon thematic cartography

One of the most common thematic map types is the choropleth, where individual areas are shaded or coloured according to statistical value. It is often said that you should not use choropleth maps for count data – the reason being that large areas often have a lot of data in them (you are likely to find more trees in a large forest than in a small one). The point is demonstrated by the following three ‘by area’ maps, where the count is square kilometres.

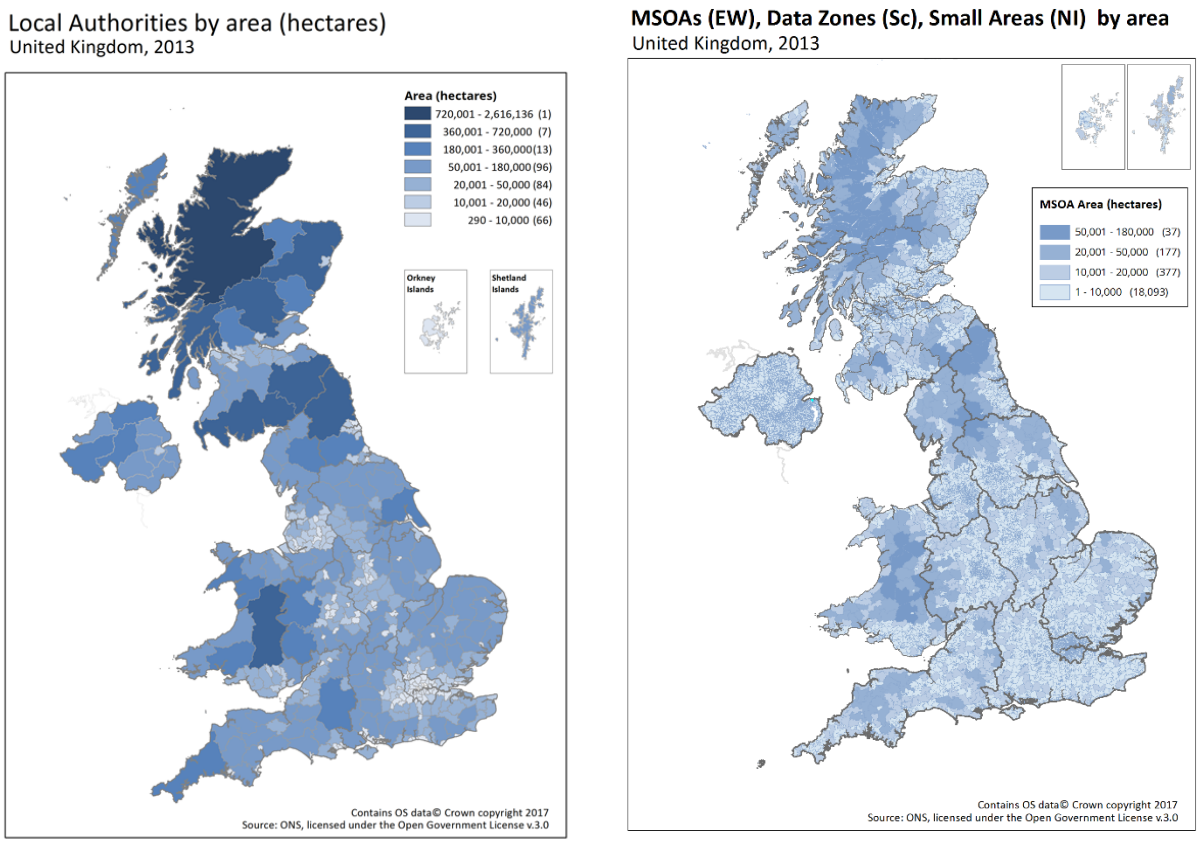


Figure 1 UK LADs and MSOAs by area

One is usually encouraged to use rate data for choropleths instead (e.g. percentage of one species of tree in the forest) as that were the perfect solution. But the tyranny of large areas persists regardless of your data type – a larger forest is more visible.

There are ways around this: one can provide larger-scale (blown-up) inset maps of selected areas alongside a national map. This moderates the problem but does not solve it. Indeed, small areas that are not selected for inset mapping may be pushed further into the background.

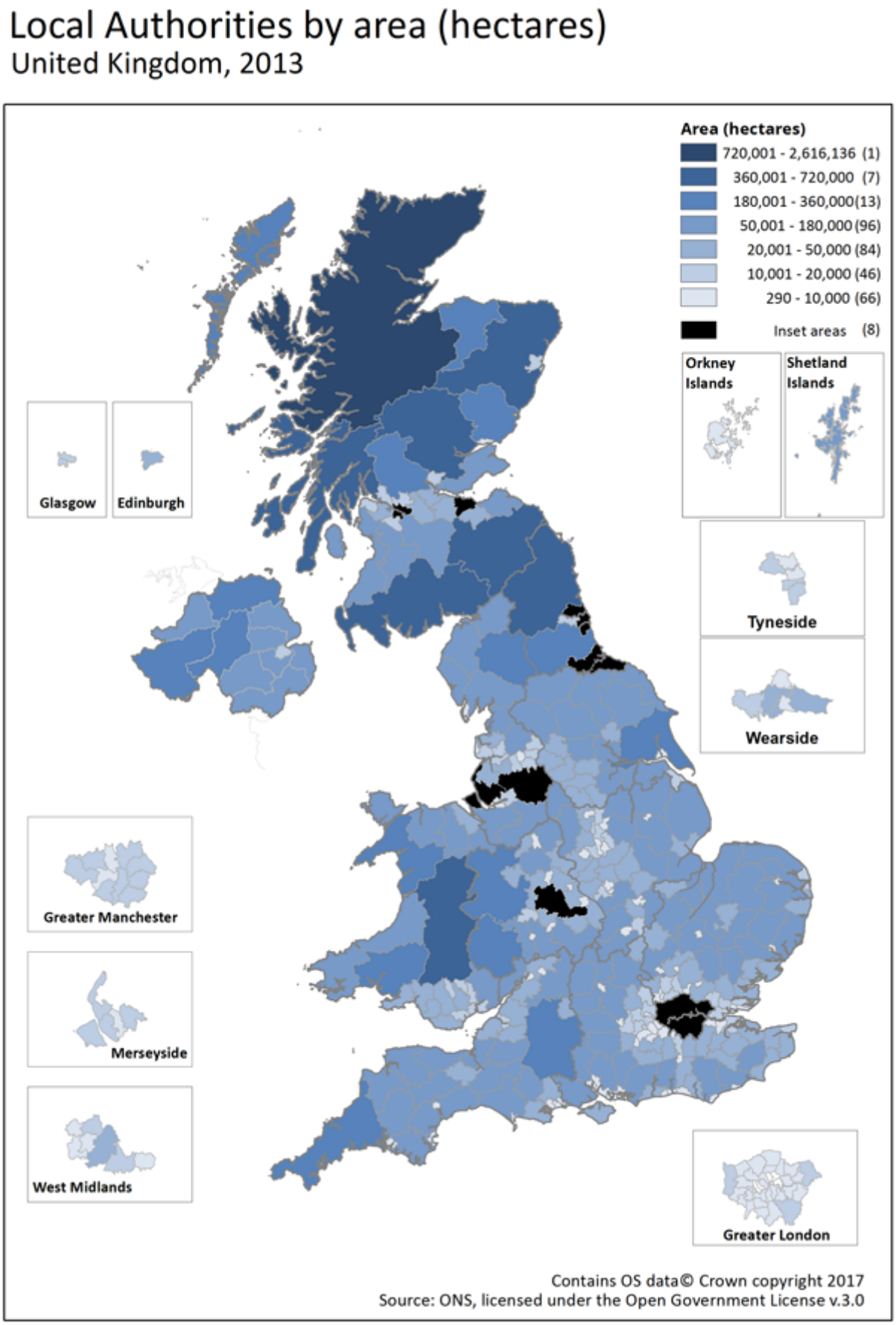


Figure 2 UK LADs by area with insets

Cartograms

But it is possible to set aside the visual dominance of large areas altogether. One can scale the territories not according to their actual geographic area, but according to a value of statistical interest. This is known as a cartogram – a hybrid statistical graph / map, a compromise by which the viewer gains more immediate access to the data but loses some of the geographical context.

Cartograms exist in many forms; they may preserve, or ignore, shape, contiguity and orientation. They are often shown together with a conventional map to aid navigation and interpretation. They can powerfully convey general impressions (see the examples below from Benjamin Henning's 'WorldMapper' [1]), but it's hard to derive specific values, and next to impossible to assess whether differently shaped areas represent the same statistical value. Also, a new cartogram must be drawn for each new variable, refreshing the navigational challenge every time.

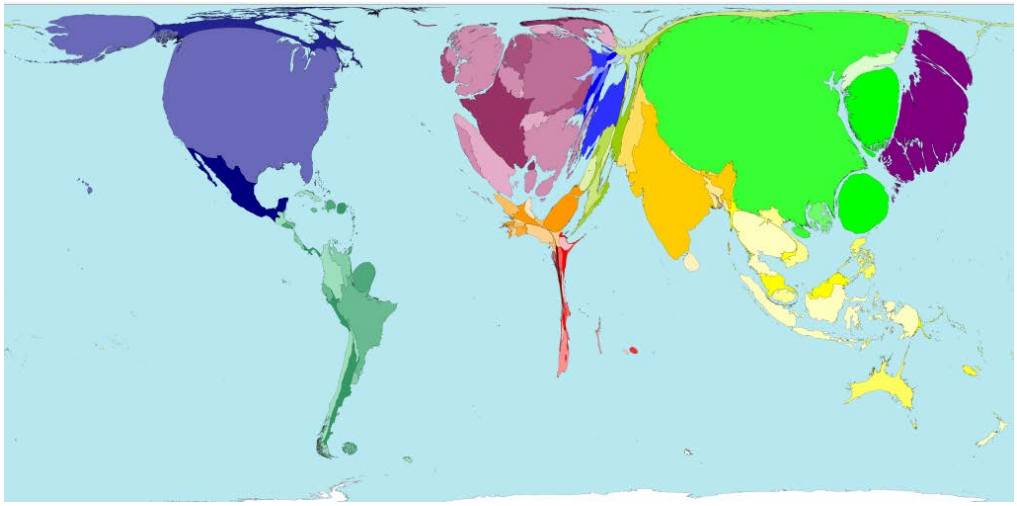


Figure 3: Wealth, 2015

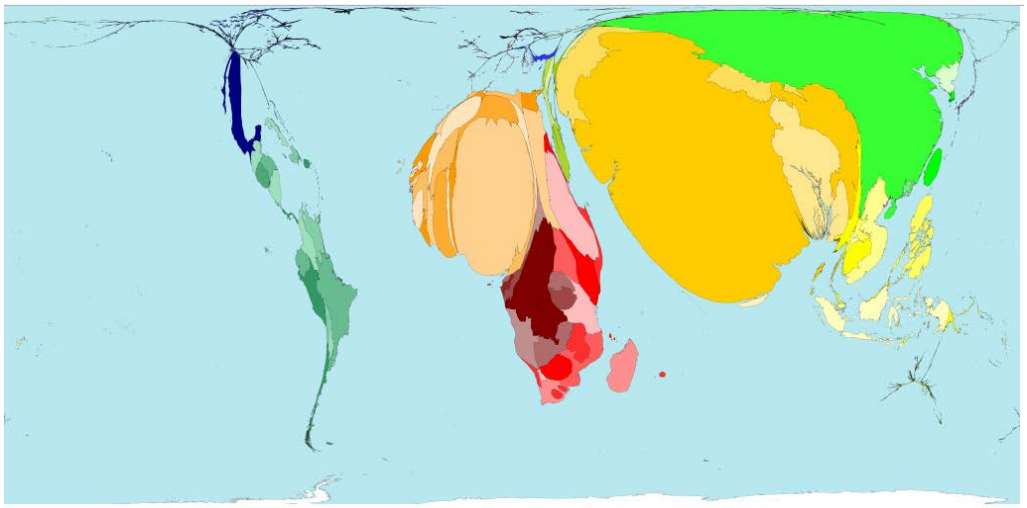


Figure 4: The Wretched Dollar (up to $1 a day)

[1] http://www.worldmapper.org/

The Equal Area Cartogram

Our favoured approach is the equal area cartogram (EAC). Here, every district is represented with an identical geometric object of equal size and shape, and therefore, area. No area is inherently (dis)advantaged, as the base geometric representation of all is the same. Once generated, the EAC provides a single standard layout that can be linked to countless datasets to generate as many thematic maps as desired.

The ONS hexmapping tool

Within this project, we have concentrated upon automatic hexmap generation using Python. Our code runs quickly and can operate on any polygon dataset at any scale.

We have tested the product on a wide range of distinctive and challenging geographies. These include countries with predominantly north-south extent (Chile, Italy, Norway), countries with east-west extent (Russia, USA), irregular shapes (Croatia), exclaves (Russia's Kaliningrad), doughnuts (Germany - Berlin within Brandenburg), complex archipelagos (Indonesia). It works on all, regardless of projection or scale. Indeed, it works equally well on all the stars within ten parsecs of Sol (our sun) and a tiny detail from an electron microscope image of the brain of a fly.

As your input geography is imported, its projection is noted: it is then turned into pure geometry for the hexmap creation and then re-exported back to its original projection. Two hexmap variants are available. The **basic** variant overlays the original input geography and the two may be used together. The **compressed** hexmap is designed to stand alone, without the original geography or a base map. However, because it is in the same projection as the original input geography and the basic hexmap, you may overlay them for testing purposes, to assess the effectiveness of the compression.

The tool works on an **inverted gravity model** (HexTool__Compressed.py - PART FIVE: THE INVERTED GRAVITY MODEL (COMPRESSION)).

Observations are drawn towards the centre of the dataset, defined by the centroid of all the areas' centroids (Centroid of Centroids (CxC)). Starting out from the CxC, the gravitational force F is negligible, but increases in power with increasing distance from the CxC. However, this produces increasing amounts of white space closer in towards the CxC, because observations towards the centre are subject to increasing friction (less antigravity) and move less and less towards the CxC.

I therefore developed the **modified inverted gravity model**, which adjusts F by reference to the difference between the distance from any observation to the CxC and the MEDIAN distance.

Speed of execution depends upon your PC's resources and the complexity of the input spatial dataset: for over ¾ of tested geographies, the compressed hexmap is created within ten seconds. Only three took more than one minute – the 1,973 Cantons of France (138s); the 1,385 NUTS3 units of the WEU and EFTA (150 s) and the 7,201 MSOAs of England and Wales (40 minutes).

Using the tool

A. PREREQUISITES

This product has been tested on Python 3.7.2 using Spyder 3.3.4 within Anaconda Navigator 1.9.7, running under 64-bit Microsoft Windows 7 and 10. [2] I do not make any promises about any other permutations.

Python libraries

- Time; numpy v1.1.6.2; matplotlib; scipy; shapely; geopandas

Installing the Python libraries

At the Anaconda prompt, enter the following commands. The stated order appears to be significant.

```
conda install -c conda-forge scipy
conda install -c conda-forge geopandas
conda update -n base conda
conda install -c conda-forge fiona
conda install -c conda-forge setuptools
conda install -c conda-forge shapely
```

Further information on these libraries:

- https://anaconda.org/conda-forge/scipy
- https://anaconda.org/conda-forge/geopandas
- https://anaconda.org/conda-forge/fiona
- https://anaconda.org/conda-forge/setuptools
- https://anaconda.org/conda-forge/shapely

The input geography must be an **ESRI shapefile** of **polygon** type. Point datasets may be accommodated by first generating buffers of dimensions appropriate to the dataset's spatial scale

Support for OGC GeoPackage (.gpkg) is being developed but has not yet been implemented.

2 NOTE: It is not possible to get this to work on a PC running within the ONS network. You must have unrestricted access to the Internet to allow the installation of the **Python libraries.**

B. SETUP

The code comprises five separate Python scripts:

B1. SETUP FILES

HexTool__Setup.py

Used to set up:

- Paths and folders
  - ROOT_FOLDER
  - Input path (INPUT_POLYGON_PATH)
  - Output paths (OUTPUT_IMAGE_PATH, OUTPUT_HEXMAP_PATH)
- Choice of geography file – UK or Elsewhere
- Hexmap type – BASIC or COMPRESSED
- Output file format (currently, only ESRI Shapefile implemented)

HexTool__Geographies_UK.py or ***HexTool__Geographies_Elsewhere.py***

Used to set up Input geography and hexmap settings

a. GEOGRAPHY_NAME (input geography)

b. HEXSIZE (width - in units of the input geography's projection) of the hexagons forming the hexmap.

c. HEXORIENTATION ('pointy-uppy' or 'flatty-uppy').

d. COMPRESSION_FACTOR for compressed hexmaps.

Note: optimal values for b, c and d are currently obtained through trial and error.

B2. CORE CODE

HexTool__Basic.py

This applies mathematical procedures to the centroids of the input geography and creates hexagons at those locations, as defined by the parameters (above).

It is recommended to generate a basic hexmap top determine the optimal HEXSIZE value before proceeding to a compressed hexmap.

B3. CORE CODE WITH COMPRESSION

HexTool__Compressed.py

This version of the code is identical apart from additional geometric sections that scale the output hexagons onto a smaller space. The distance between distant locations is reduced by a scaling factor that more powerfully affects locations more distant from the centroid of centroids (CxC) than it does locations closer in. The key consideration is that the output still retains a close similarity to the original spatial distribution of the input geography centroids.

B4. SETUP PROCEDURE

Within script "HexTool__Setup.py"

PATHS AND FOLDERS

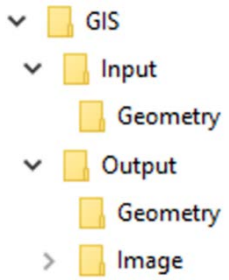
The **ROOT_FOLDER** depends upon the PC you are running the program from. Examples:

ROOT_FOLDER = r'C:/HEXMAPS/' - ONS - ONS27797

ROOT_FOLDER = r'D:/HEXMAPS/' - ONS - geoCentaurs

ROOT_FOLDER = r'G:/HEXMAPS/' - HOME - MESH II

For any other PC, create the 'HEXMAPS' ROOT_FOLDER in desired location, along with the following **sub-folders**, and add to the HexTool_Parameters.py script a line of code like those above:



The following three PATH variables depend upon the ROOT_FOLDER and folder structure as above.

INPUT_POLYGON_PATH = ROOT_FOLDER + 'GIS/Input/Geometry/'

OUTPUT_IMAGE_PATH = ROOT_FOLDER + 'GIS/Output/Image/'

OUTPUT_HEXMAP_PATH = ROOT_FOLDER + 'GIS/Output/Geometry/'

GEOGRAPHY

Activate **either** 'import HexTool__Geographies_UK as geographies'

or

'import HexTool__Geographies_Elsewhere as geographies'

HEXMAP TYPE

Activate **either** 'import HexTool__Basic'

or

''import HexTool__Compressed'

OUTPUT FILE TYPE

This is currently set to ESRI shapefiles (shp).

OGC GeoPackage(gpkg) and GeoJSON output can also be specified, but this has not been implemented.

B5. SETUP PROCEDURE – FOR A BASIC HEXMAP

Within script "HexTool__Geographies_UK.py" or "HexTool__Geographies_Elsewhere.py"

INPUT POLYGON and PARAMETERS

1. Choose your GEOGRAPHY_NAME. Ensure that this is present – as a shapefile – in the INPUT_POLYGON_PATH folder.
2. Select a value for HEXSIZE. The optimal value subject to taste, but dependent upon:
    a. The distance units of the input geography polygon shapefile's projection (e.g. decimal degrees, miles, meters, microns).
    b. The E-W and N-S extents, in those distance units, of the input geography.
    c. The number of polygons in the input geography.
3. Set HEXORIENTATION to 1 (vertical, or 'pointy-uppy') – generally more suitable for geographies with an EAST-WEST extent (Greater London), or to 2 (horizontal, or 'flatty-uppy') - for geographies with NORTH-SOUTH extent (the UK).
4. There must be a value for COMPRESSION_FACTOR, even if it not called. Default value '3'.

B6. SETUP PROCEDURE – FOR A COMPRESSED HEXMAP

Within script "HexTool__Geographies_UK.py" or "HexTool__Geographies_Elsewhere.py"

Follow steps 1 – 3 as above.

1. Set COMPRESSION_FACTOR to an appropriate value.
    a. Logically permissible values extend from 1,000 (almost no perceptible compression) to 1 (extreme compression). Reliable results can usually be obtained at CF = 1.2.
    b. The optimal value produces the smallest, most compact version of the original pattern that is still recognisable as such. It is up to the mapmaker to decide when this has been achieved.

RETURN TO AND RUN script "HexTool__Setup.py"

Examples

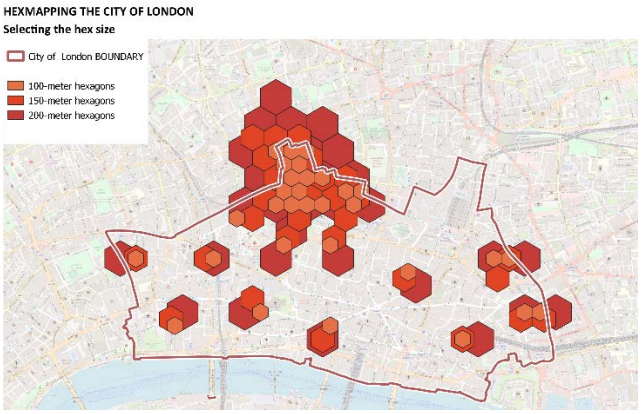


Figure 5 Variations on HEXSIZE - The City of London

Here, several versions of the **basic** hexmap have been generated, each with a different hex-size. Here, the results for 100, 150 and 200 meters are shown. HEXSIZE = 100 meters produces a basic hexmap where all hex centroids are within the original boundary polygon.

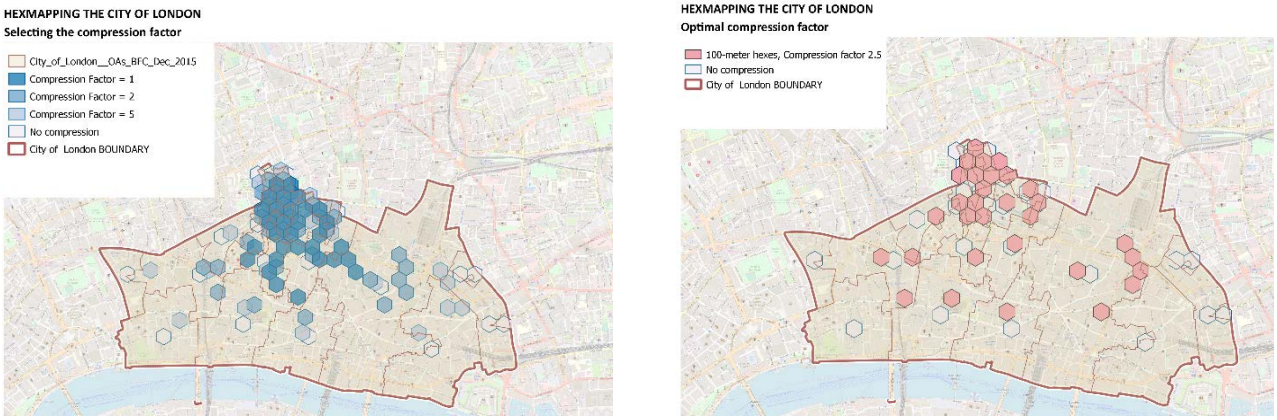


Figure 6 Variations on COMPRESSION_FACTOR - The City of London

Retaining the parameter values from the **basic** hexmap, several compressed hexmaps are now created, with varying COMPRESSION_FACTOR values applied to the 100-meter hexagons. The user decides which result is most effective.

The Compression Algorithm

The compression algorithm works most effectively on heterogeneous geographies, comprising urban concentrations together with large sparsely-populated expanses, such as Canada and Australia.

As we have already seen, a national-level choropleth map of such a geography will be dominated by large, relatively empty, areas, while urban areas, which tend to be relatively small, may be all but invisible. Yet socio-economic data is likely to be concentrated in the smaller areas.

If every area is represented an equally-sized symbol, the visual impact of each district will be equalised. On a basic hexmap, each hexagon is rooted to the geometric centroid of its district: so, the hexmap can *accurately overlay on a basemap and the* geographical distribution is faithfully represented.

However, this will leave large gaps between the symbols due to the distances involved, so again, the large rural areas will dominate.

Figure 7 Districts of Canada / Australia - basic hexmap

The optimal size for the symbols is where hex-clusters (representing urban concentrations) reflect the actual spatial distribution of the urban areas. Too small, and the hexes will be far apart even in the urban areas. Too large, and the urban hexes will jostle each other out of place.

However, with geographies like Canada or Australia, the optimal hex size as defined above will produce an enormous amount of empty space. This results in small hexagons that are difficult to label and interpret.

In such cases the compressed hexmap comes into its own. This is built on the optimal hex size that we have determined above.

We calculate the centroid of centroids (CxC) – the geographical mid-point of the centroids of all the areas. All centroids are then shifted towards the CxC on the basis of the Compression Factor (CF). described above. The CF is analogous to gravitational attraction, and various models were pursued to explore shades of positive and inverse gravity fields.

As the Compression Factor is adjusted (remember that CF=1,000 is practically no compression, while 1 is extreme compression), the gravitational attraction of the CxC varies. Centroids for peripheral areas will have the furthest to travel towards the CxC.

The user will determine that, at a certain CF value, the pattern of hexes remains recognisably that of the districts they began with. Look out for – and seek to emulate – particular patterns and concentration in the original boundary set

You will now have a hexmap layer (ESR Shapefile) that you can use independently of a basemap and to which you can join ant desired attribute data. Zoom to layer and the hexagons will be distinctly larger and their labelling clearer than would have been possible with a basic hexmap.

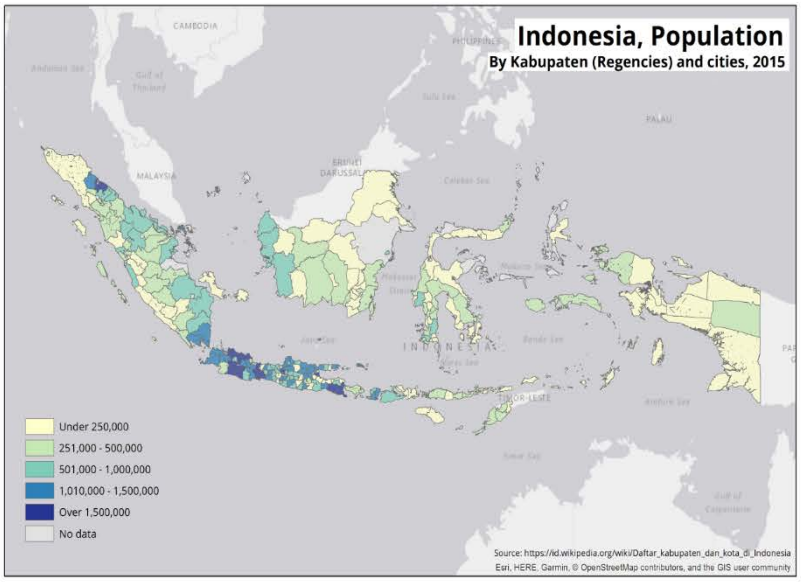
Complex archipelagos

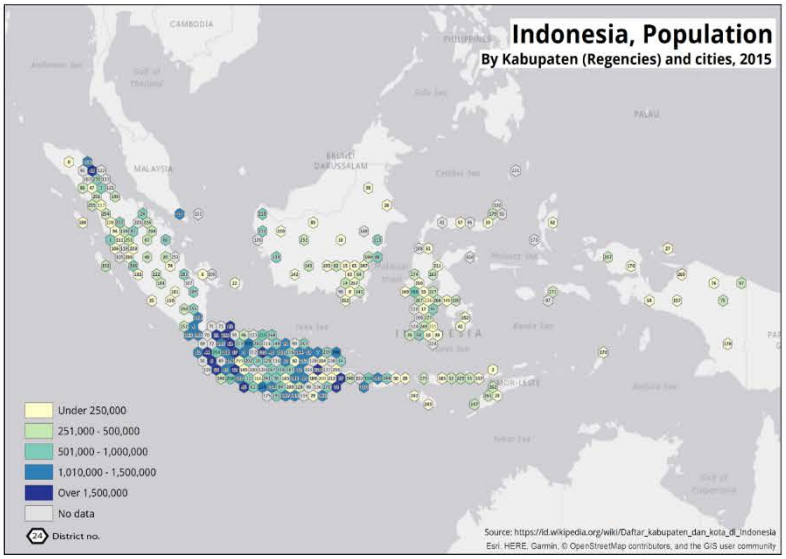
The program works well with highly complex and regionally differentiated geographies such as Indonesia.

Here, 270 Kabupaten or ‘regencies’ are distributed across a huge number of islands. There is considerable variation in size, generally being large on the islands of Borneo and New Guinea, but tiny on the densely inhabited island of Java. This is a classic example of the shortcomings of choropleth mapping. In terms of mapping socio-economic data at this administrative level, Java will always be disadvantaged.

A basic hexmap (HEXSIZE = 0.6 decimal degrees) will result in the concentration of regencies on Java expanding well beyond the island’s coast, while great spaces exist between the hexes representing the extensive regencies on Borneo and New Guinea. The HEXSIZE required to accommodate this diverse geography means that effective labelling is very challenging.

But a compressed hexmap., with a COMPRESSION FACTOR of CF=3, results in the large gaps being reduced while the overall pattern is clearly retained. One can then zoom into the map and label it more effectively.





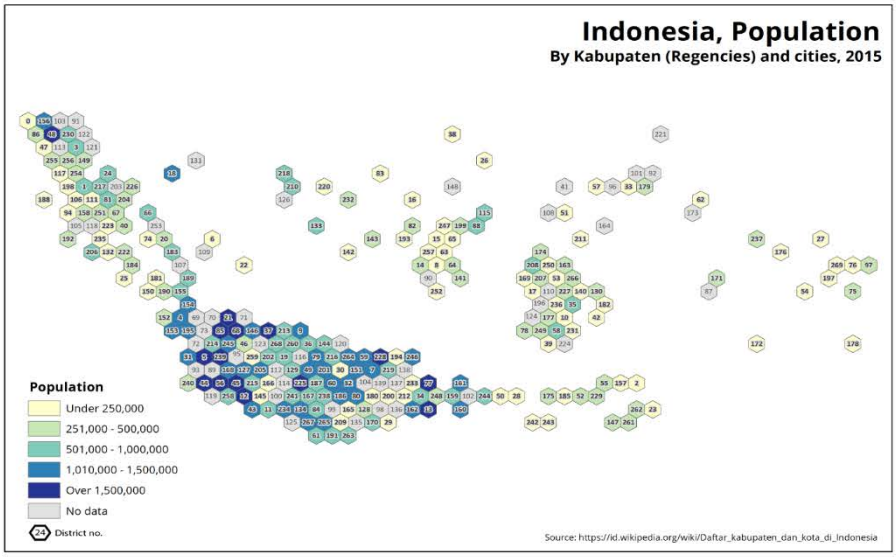


Figure 8 Regencies of Indonesia, basic and compressed hexmap

Applications to the UK

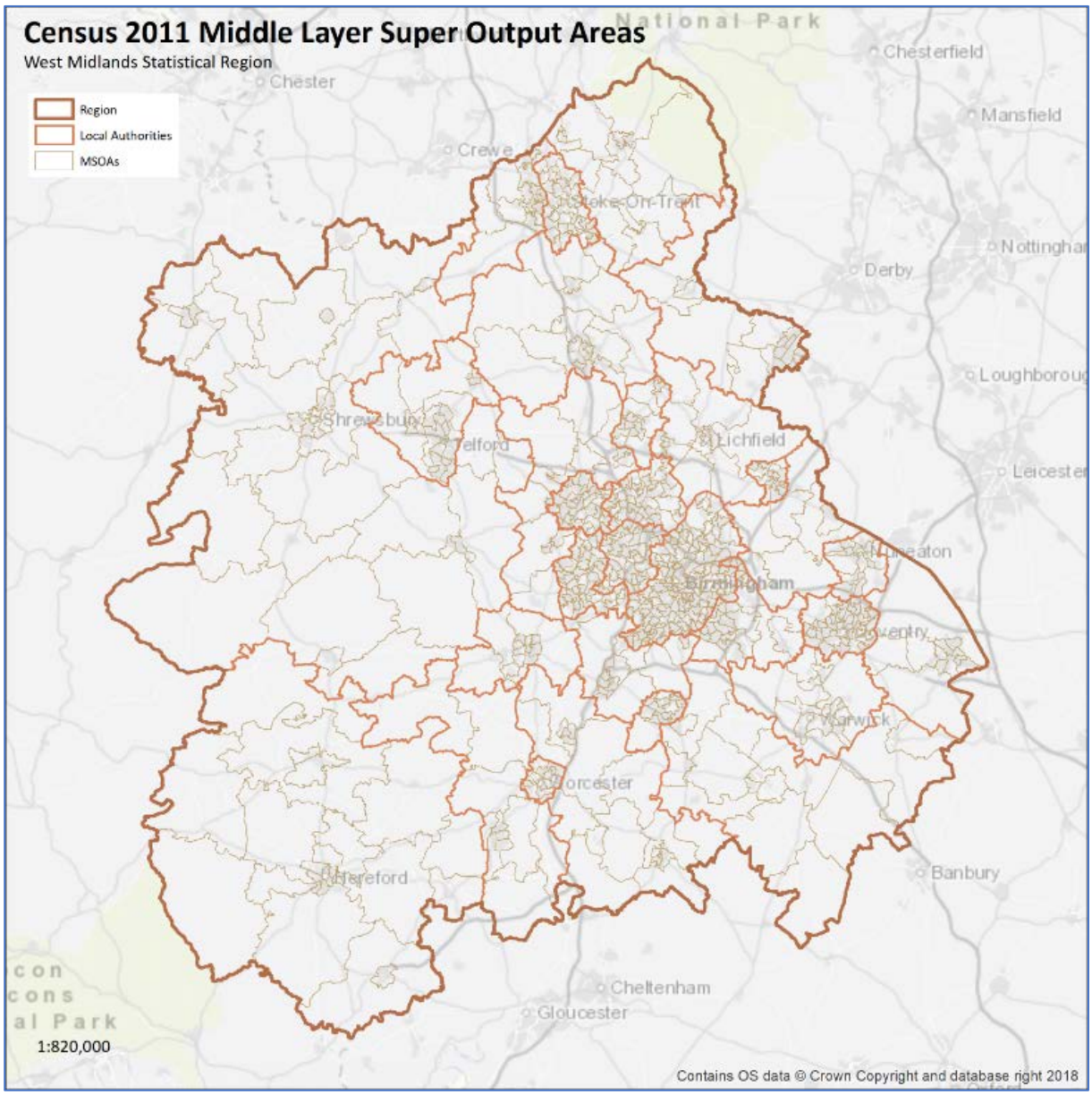


Figure 8 MSOAs in the West Midlands Statistical Region

The West Midlands region encompasses some to of the most rural and some of the most urban parts of England, giving rise to a 700-fold size difference between smallest and largest of the 735 MSOAs in the region. It is therefore not practicable to produce an MSOA-level choropleth, graduated or proportional symbol, or basic hexmap of the entire region (scale 1:820,000).

On the other hand, a compressed hexmap can succeed, as the compression permits a zoomed-in view, equating to a scale of 1:475,000.

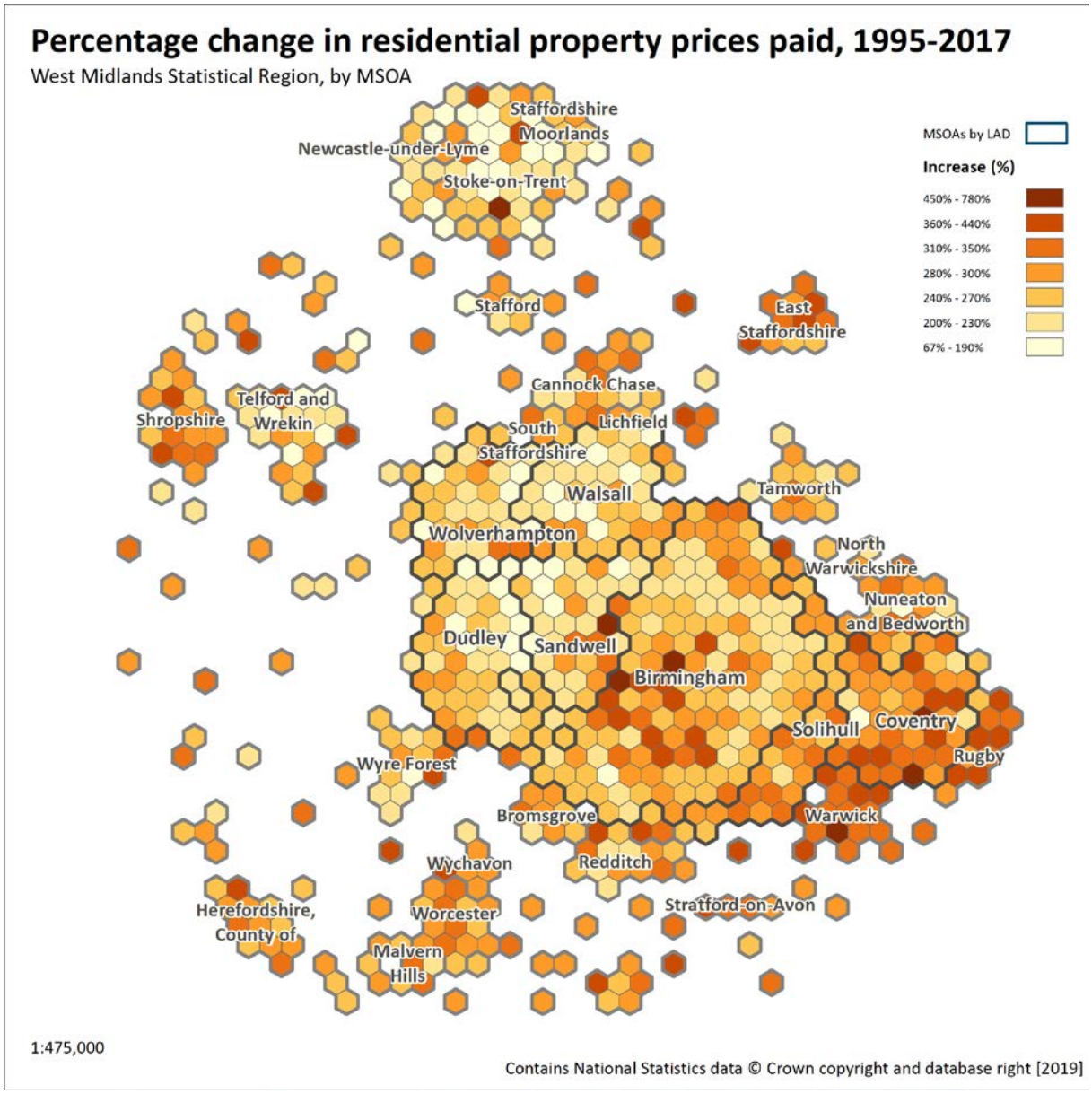


Figure 9 MSOAs in the West Midlands Statistical Region – compressed thematic hexmap

Inset hexmaps

Where clusters of spatial units could cause unreasonable distortions of the overall pattern of the hexmap, these can be extracted to inset maps alongside the general map. In contrast with the earlier example, because every area is represented by an identical symbol, areas not selected for inset mapping are not disadvantaged.

Furthermore, generating individual hexmaps for the inset areas results in a better approximation to their actual shape than can be achieved by creating a single overall hexmap. Both basic and compressed hexmap variants (with the inset areas extracted) can more accurately display the areas around the periphery of the inset areas (see around London).

Deaths of homeless people (estimated), in 2017

Local Authority Districts (LADs), 2017, England and Wales

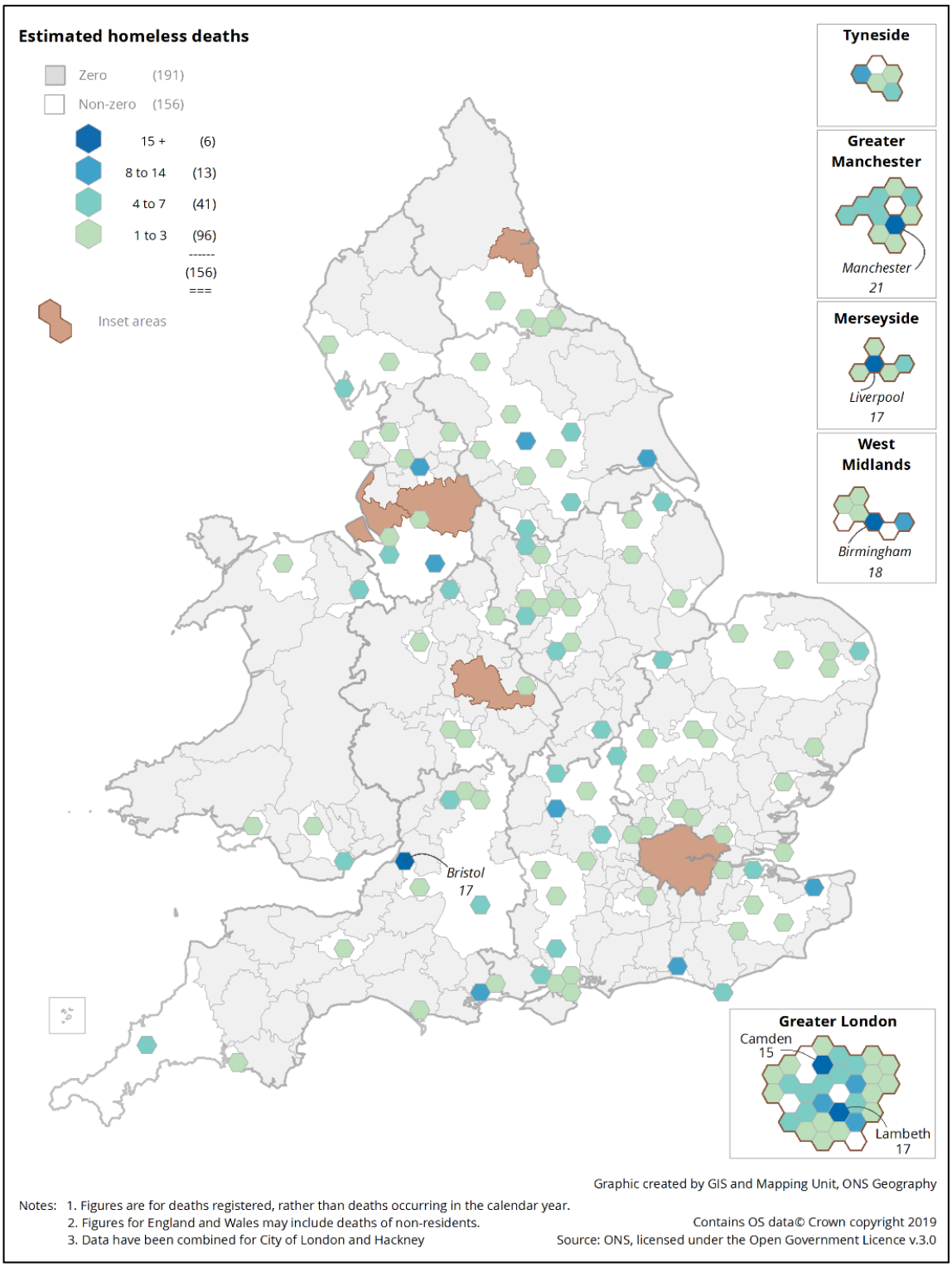


Figure 10 Deaths of Homeless People (est.) 2017. Basic hexmap.

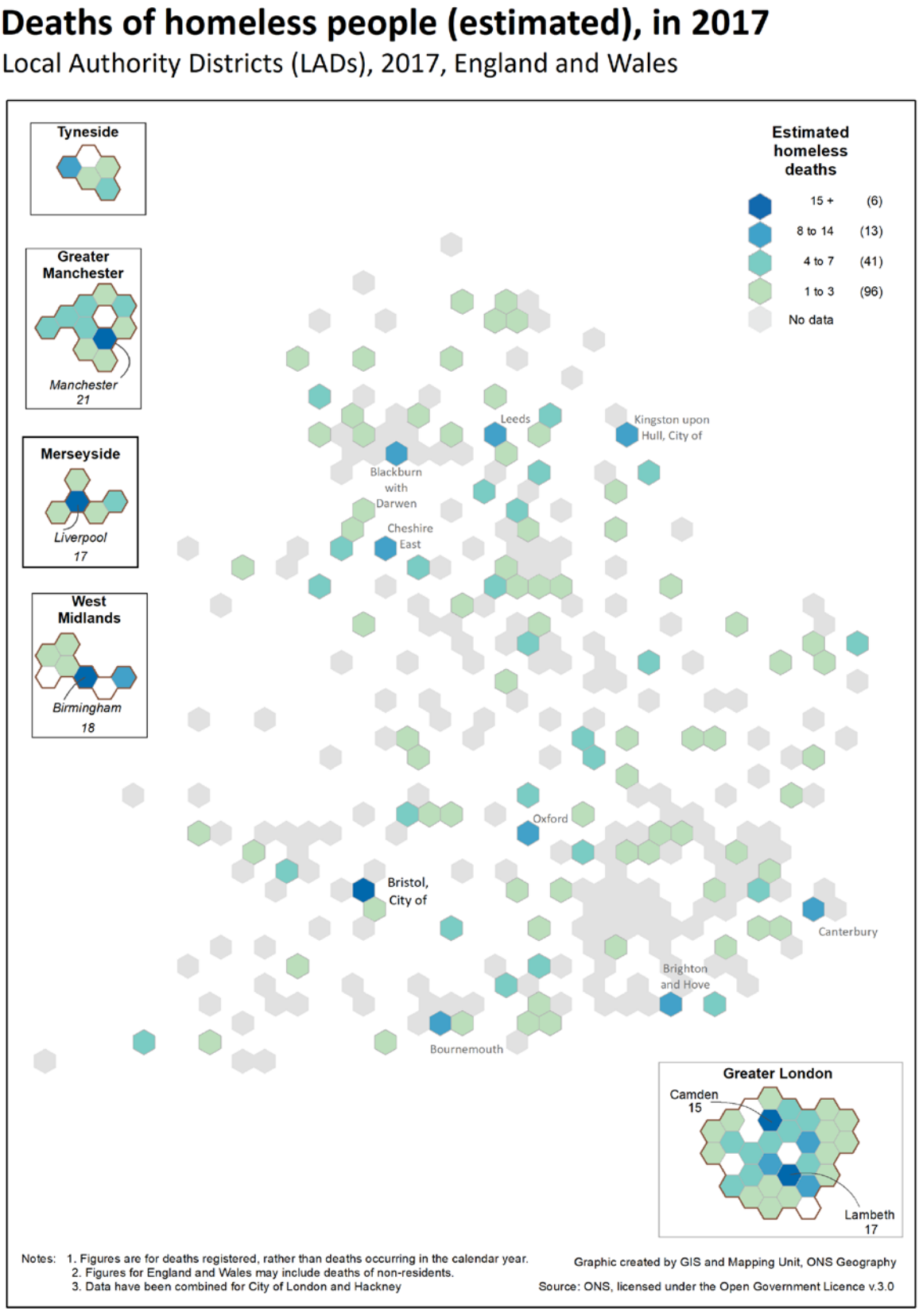


Figure 11: Deaths of Homeless People (est.) 2017. Compressed hexmap.

Regional and multiple compression

For some hierarchical geographies (e.g. Local Authorities within regions within countries) with distinct spatial patterns, and where it may be important to retain these, it may be desirable to initially generate regional hexmaps before combining these into a hexmap for the whole geography. Projection, HEXSIZE and other parameters must be common across all regions.

Each regional hexmap will have its own CxC drawing the individual hexes inwards, opening gaps between regions. The datasets may be merged into a single shapefile, allowing the gaps to be removed by shifting the hexes of one region towards the other, respecting the common mesh.

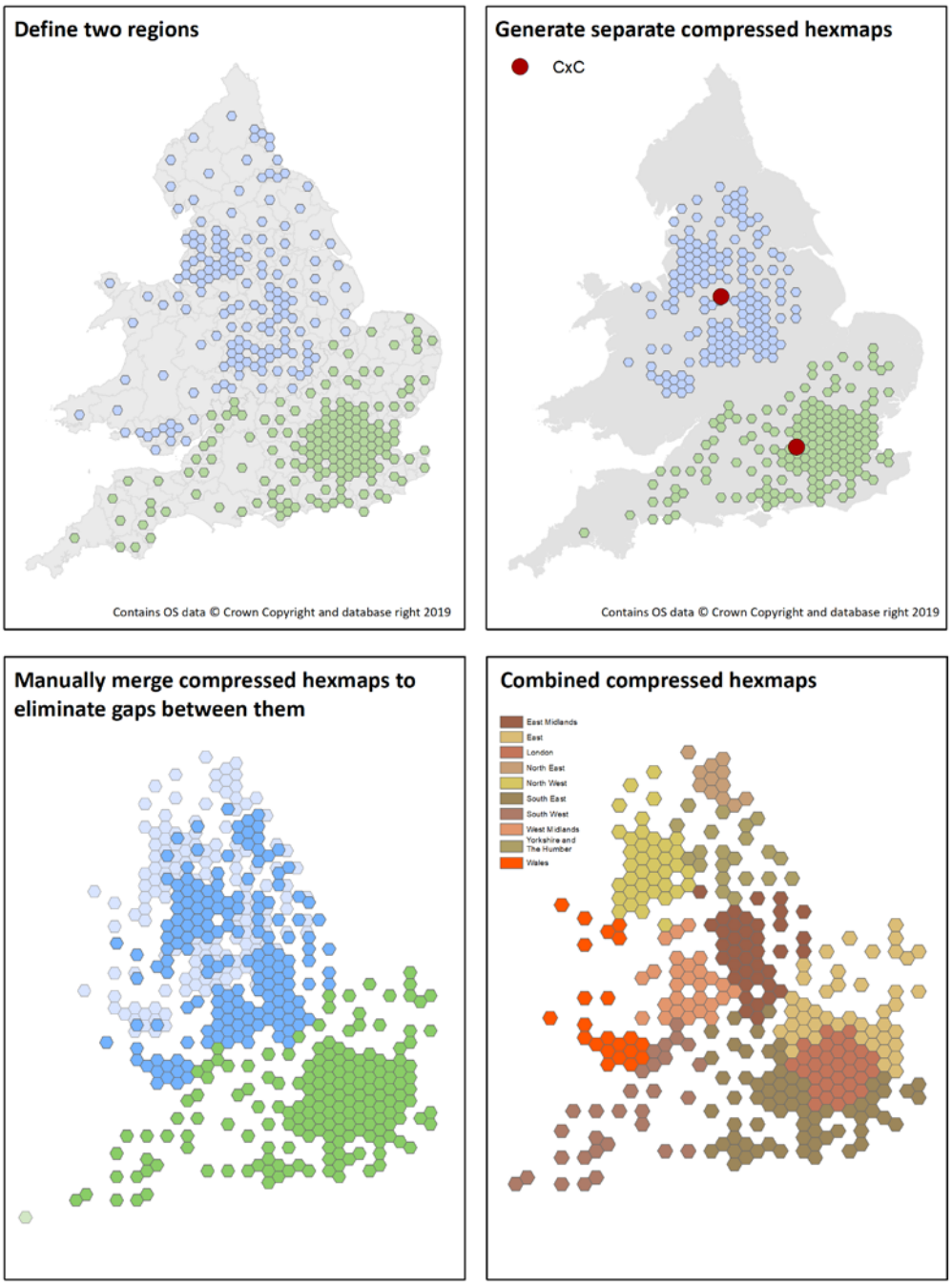


Figure 12 Merging two regional compressed hexmaps into one

Other scales

Local

The methodology may have applicability well beyond the scales normally used for cartography, and well beyond social geostatistics. Whether the application of this techniques to data at these scales is helpful or even appropriate, must be left to the relevant subject experts.

Two datasets have been kindly provided by Dr Benjamin Ciotti of the University of Plymouth's School of Marine Biology. The first is a set of marine biology sample points in a small part of Plymouth harbour. The x-y location of each sampled point was recorded along with depth and data on the various species found there. The second is a selection of sampled beaches on the western coast of Scotland. In both cases, the dispersal and clustering of the sampling points makes the use of proportional symbols in their actual locations inefficient, with both large gaps and overlaps. In both cases, compressed hexmap are more effective.

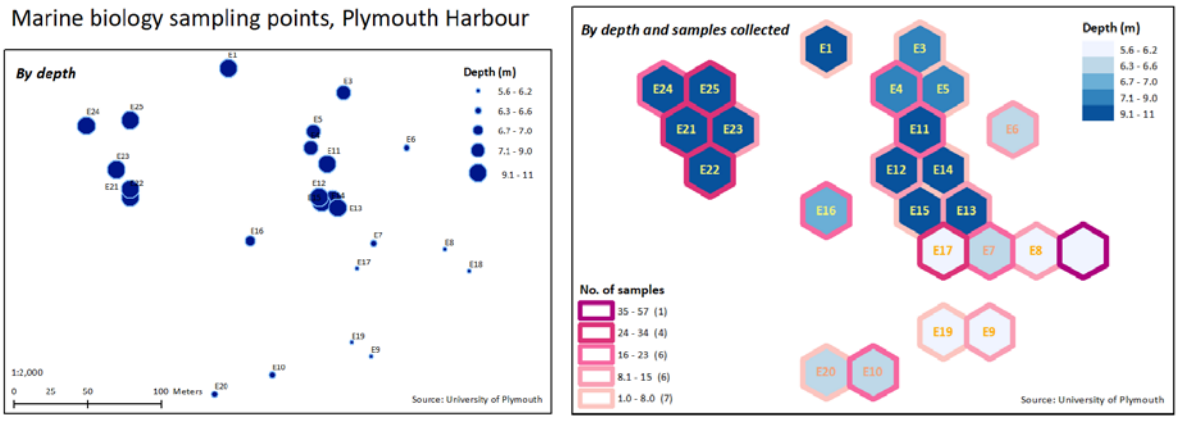


Figure 13 Plymouth Harbour

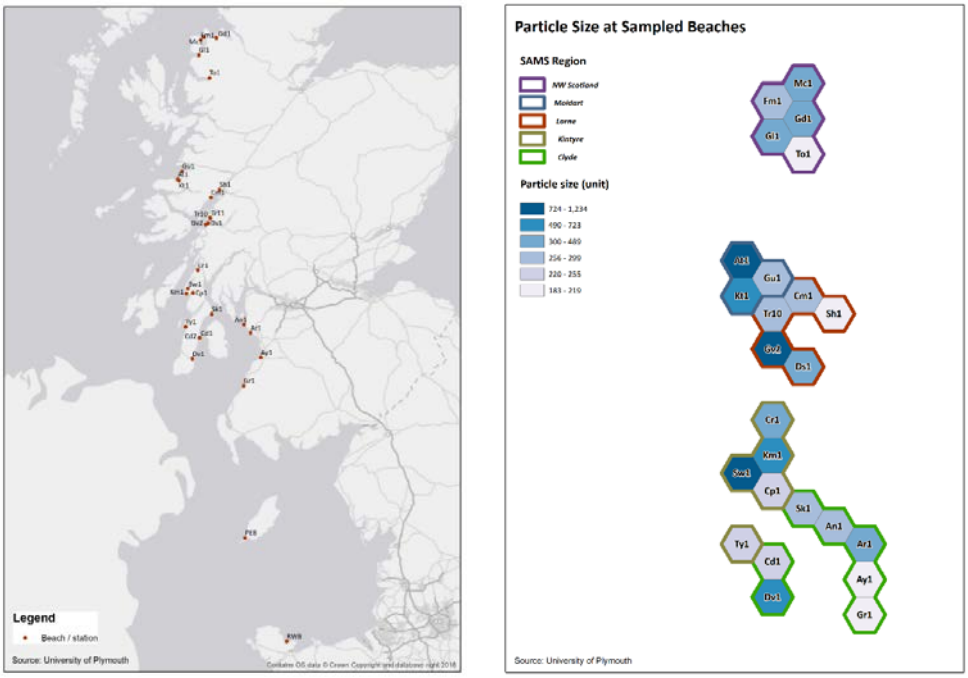


Figure 14 Scottish beaches

Microscopic

Zooming right in, we have generated a (proof-of-concept) compressed hexmap from an electron microscope image of the brain of a fly. In the absence of spatial data on the image and purely as a proof of concept, I converted the image file to vectors and applied a simple Cartesian coordinate system. I then created and mapped a simple (and meaningless) 8-value categorical scale on a small part of the image which, at 1:1200, equates to 0.000025 mm^2.

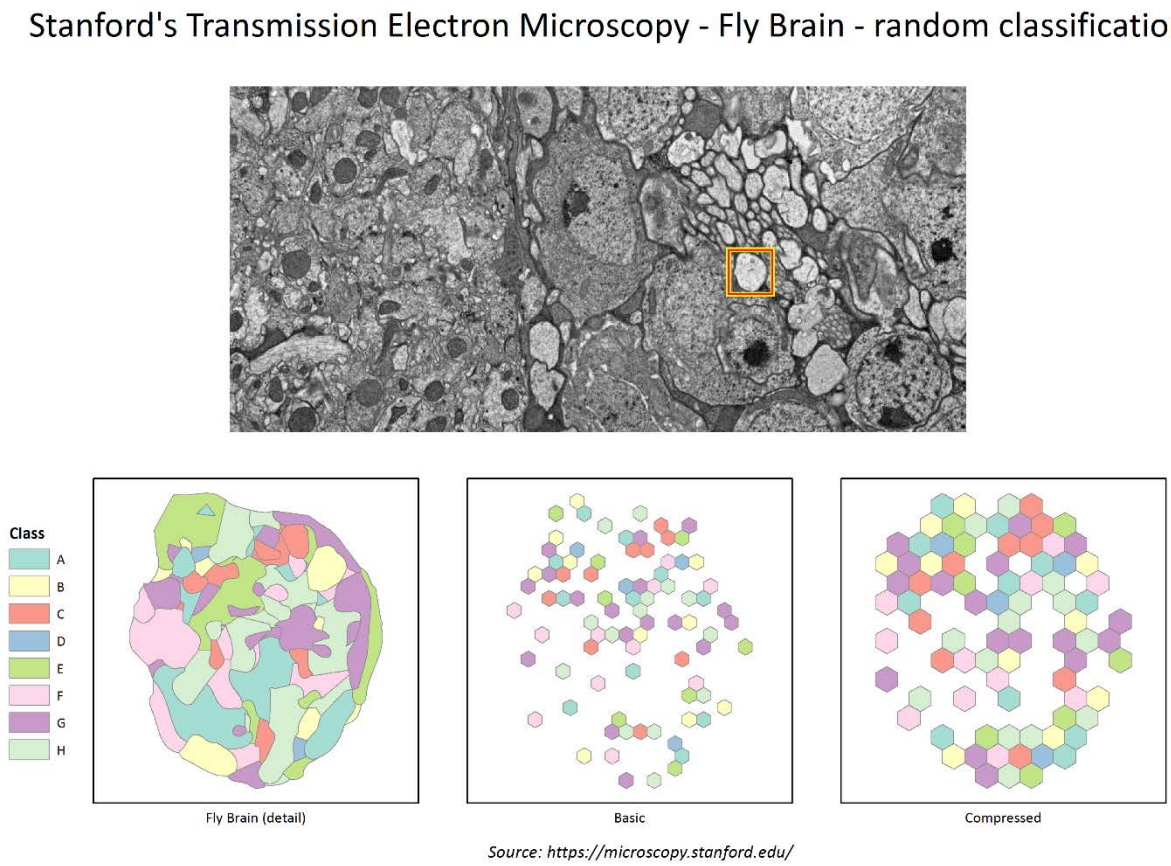


Figure 15: Transmission Electron Microscopy – Fly brain at x1200 magnification (detail)[3]

3 https://microscopy.stanford.edu/

Cosmic

Looking further afield, the technique may also be used at the cosmic scale. The HYG Database is a meta-catalogue containing over 120,000 stars. It includes the x,y,z, Cartesian coordinates of the star (in a system based on the equatorial coordinates as seen from Earth), distance from earth and a range of other fields. Once a suitable buffer is generated for each point to create polygons, a compressed hexmap may easily be generated for any subset of the catalogue. We have generated a compressed hexmap of all the stars within ten parsecs of Sol (our sun). [4]

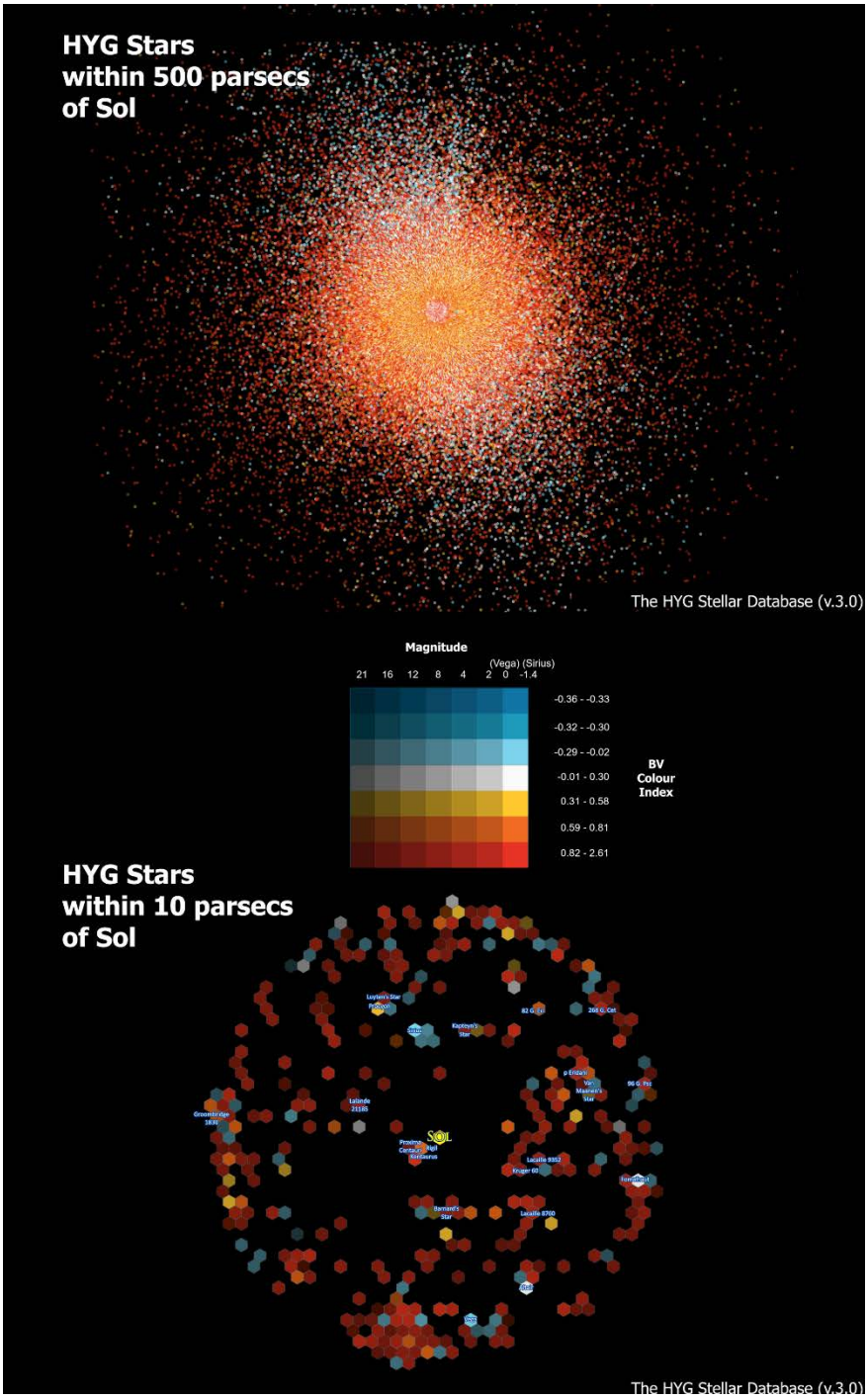


Figure 16 Stars within ten parsecs of Sol

4 One parsec = 3.26 light years. The Milky Way galaxy is 30 kiloparsecs (kpc) across, and Sol is 8kpc from the galactic centre.

Summary

We have created a tool that can be used to quickly and efficiently create hex-based equal area cartograms from any dataset that includes cartesian or projected coordinates, regardless of the scale.

It is most suitable for datasets where the area of polygons, or the distances between points, is highly variable.

The script requires a polygon input. A point dataset must first be converted to polygon by generation of buffers.

Your polygon dataset must then be placed in the INPUT_POLYGON folder, whose location is defined in script 'HexTool__Setup.py'. It must also be added to 'HexTool__Geographies.py' as GEOGRAPHY_NAME.

Your polygon input dataset must be placed in the INPUT_POLYGON folder, whose location is defined in script 'HexTool__Setup.py'. It must also be added to 'HexTool__Geographies.py' as GEOGRAPHY_NAME. You also have to select values for HEXSIZE (in units of the projection), HEXORIENTATION (1 or 2) and COMPRESSION_FACTOR (sensible range 1.2 -> 4).

Note that a value is required for COMPRESSION_FACTOR even if you are creating a basic hexmap. Use '2' as the default. Geometry and image files will be generated and placed in the output folders specified.

Within 'HexTool__Setup.py', select **basic** or **compressed** hexmap.

Run 'HexTool__Setup.py'.

Geometry and image files will be generated and placed in the output folders specified.

Play with the HEXSIZE, HEXORIENTATION and COMPRESSION_FACTOR parameters until you are satisfied.

Note: if you use a decimal for the COMPRESSION_FACTOR, e.g.2.5, while the folder for the output will be named correctly (e.g. ZAF_adm2_mainland__0.4_1__CF3___C_HXMP), the name of the shapefile within it will be curtailed at the decimal point (e.g. ZAF_adm2_mainland__0). If you try several options (e.g. CF 2.8, 2.6, 2.4) they will all be in separate, correctly named folders, but will have identical filenames.

Recommended testing:

First: try out some of the geographies supplied with this tool with the parameters as set up in the Python script 'HexTool__Geographies_UK.py' or 'HexTool__Geographies_Elsewhere.py'.

Then see what happens when you alter HEXSIZE, HEXORIENTATION and COMPRESSION_FACTOR.

Add some of your own data (as ESRI shapefile) to the INPUT_POLYGON_PATH folder and to the appropriate 'HexTool__Geographies_....py' script. Experiment with the three parameters.

DESIRABLE OPTIMISATIONS

- An outline of the original geography would assist orientation. Unfortunately, the compression algorithm operates on hexagons and not the original polygons.
- Currently, the optimal values for hex size, hex orientation and the compression factor must all be obtained by trial and error. It should be possible to calculate optimal values by reference to the perimeter envelope of the input geography, the number of spatial entities within it and the units of the spatial reference system (projection).
- The tool should be able to accept as input any spatial file format recognised by Fiona, but it's currently set up only to accept ESRI shapefiles as input and to write output also only to shapefile. Adding other formats at both ends (e.g. geopackage, geoJSON) – has been experimented with but not implemented.
- The basic and compressed code began as a single script, remain essentially the same and should really be merged back into each other, incorporating a conditional IF COMPRESSED HEXMAP – ACTIVATE COMPRESSION CODE – ELSE SKIP clause. Otherwise there's double the maintenance. A 'compressed' switch should be added to the parameters.
- Some of the linear code ought to be rewritten as functions
- As a cosmetic touch, perhaps a completion bar / hourglass / sound on completion could be added.
- Under some circumstances, multithreading would help.

ERRORS

On occasion, the script does not fire up. Simply run **HexTool__Setup.py** again and it will.

Remember to run the tool from **HexTool__Setup.py** only. Running the other scripts will not produce any output.

ATTRIBUTEERROR: 'FLOAT' OBJECT HAS NO ATTRIBUTE '__GEO_INTERFACE__'

CAUSE: GEOGRAPHY_NAME has an unknown projection, or HEXSIZE too small.
Solutions: choose and apply appropriate projection for the geography, change HEXSIZE

DRIVERERROR: PATH/FILENAME.SHP: NO SUCH FILE OR DIRECTORY.

CAUSE: typos. Can also occur if you use multiple Python interpreters on the scripts
Solutions: check you typing. Stick with one interpreter!

VALUEERROR: THE SECOND INPUT GEOMETRY IS EMPTY

CAUSE: The HEXSIZE value is not appropriate for the combination of geography extent and geometry unit. Example: global decimal degree extent -180 to + 180 but HEXSIZE = 60000
Solutions: change HEXSIZE to appropriate scale. Check the axes on the INPUT_POLYGON.png to see the min/max x and y. HEXSIZE should be something like the min/max RANGE divided by the number of map units.

VALUEERROR: THE TRUTH VALUE OF AN ARRAY WITH MORE THAN ONE ELEMENT IS AMBIGUOUS. USE A.ANY() OR A.ALL()

CAUSE: failing to delete variables at the end of the scripts.

'MAXIMUM ALLOWED SIZE EXCEEDED'

CAUSE: COMPRESSION_FACTOR = 0. Must be at least 1. Min 1.2 recommended.

ATTRIBUTEERROR: MODULE 'HEXTOOL__SETUP' HAS NO ATTRIBUTE 'INPUT_POLYGON_PATH'

This error paired with occasional double-running of the script. It's to do with:
import HexTool__core_coreOnly.py and **import HexTool__core_withTransform.py.**

These scripts both '**import HexTool__Setup as parameters'** and

```
INPUT_POLYGON_PATH = parameters.INPUT_POLYGON_PATH
OUTPUT_IMAGE_PATH = parameters.OUTPUT_IMAGE_PATH
OUTPUT_HEXMAP_PATH = parameters.OUTPUT_HEXMAP_PATH
OUTPUT_FILE_TYPE = parameters.OUTPUT_FILE_TYPE
```

… but both are run from **HexTool__Setup.py**