## About the AddressBase mapping tool

This mapping tool takes AddressBase data, plots it on a map, and shades underlying maps based on the number of those points in each area on the map. It outputs a leaflet map which includes layer controls and shows relevant data on hover. It also calculates the number of points per 1000 population based on 2021 population figures and local authority boundaries for comparison.

The map it makes can be displayed in a browser window or saved as a html file and shared or embedded in websites.

It was made by Samantha Iacob, an apprentice data person during her Geospatial rotation. If you have any questions please email [samantha.iacob@ons.gov.uk](mailto:samantha.iacob@ons.gov.uk)

## Who it’s for

This tool is mainly for ONS staff and other civil servants but could be useful to anyone with access to the [OS AddressBase Premium](https://www.ordnancesurvey.co.uk/products/addressbase-premium) product.

It’s designed to give a first glance at the data, highlighting geographies that could use further investigation, without a lot of work.

It’s also for people who are interested in exploring their data with maps but don’t know how to use geospatial packages. The tool is designed to be user-friendly and can quickly make interactive maps without any special coding knowledge, though as it is a Jupyter notebook it would be useful to know the basics of that software.

## Who it’s not for

Anyone looking to generate authoritative public-facing maps. To keep the data preparation simple and to allow the tool to be agnostic about what geographic boundaries it can accept, it assigns plot points to boundaries using an intersecting spatial join rather than National Statistics lookup tables. As these boundaries can be slightly incorrect or even overlap due to simplification it means that the tool may sometimes assign points to the wrong geometry, especially on borders. While this is not expected to lead to wildly incorrect results it can’t be assumed to be 100% accurate either.

## Recommended training

These are optional though it would be good to understand how Jupyter notebooks works at a minimum before using this tool.

[Python - Jupyter Notebooks Overview](https://learninghub.ons.gov.uk/course/view.php?id=637) - ONS via Learning Hub

[Jupyter notebook](https://www.dataquest.io/blog/jupyter-notebook-tutorial/) - External

[Intro to GIS in Python](https://onsgeo.github.io/geospatial-training/docs/intro_to_gis_in_python)

## Prerequisities:

A python environment with geospatial libraries as well as Pandas installed. This tool uses Pandas, GeoPandas and Folium for data preparation and mapping, and relies on sys, os and datetime for some supplemental functions.

This guide may be helpful for first time users, especially using ONS equipment:

[Installation guide for Python Geospatial packages](https://onsgeo.github.io/geospatial-training/docs/guides/python_install_anaconda)

You will also need access to OS AddressBase Premium stored on a database, either personally or through a colleague who has access and can retrieve the data you need. To access the data you will need a method of querying AddresBase, such as MS SQL Server Management Studio.

## Step by step

This is a step-by-step guide for using the tool for the first time. If you are unfamiliar with SQL or AddressBase you could ask a colleague with appropriate training and permissions to carry out these instructions for you.

### SQL instructions

Connect to your AddressBase server and create a new query. Decide on what “epoch” you are retrieving data for (epochs are AddressBase releases, one is published every six weeks). From that epoch’s BLPU table, select the following columns:

* UPRN AS uprn
* CLASSIFICATION\_CODE AS class\_code
* X\_COORDINATE AS x
* Y\_COORDINATE AS y

Left join on that same epoch’s classification table on UPRN

Using WHERE statements:

* Filter to select only the CLASSIFICATION\_CODES you want using IN or LIKE statements (a full list of classification codes is available from Ordnance Survey from its [downloads page](https://www.ordnancesurvey.co.uk/products/addressbase-premium#technical) and a copy of this file is located in the data folder of this tool’s root directory.
* Filter on CLASS\_SCHEME = ‘AddressBase Premium Classification Scheme’
* Filter out LOGICAL\_STATUS 8 (this means the classified object is historical)
* You could also filter by country - E, J, S, W

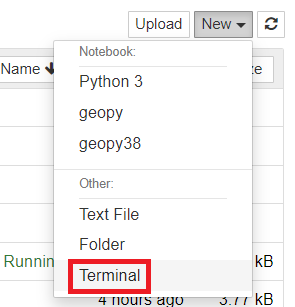
A sample snippet of SQL code used to generate the default data in this tool can be found in the *samples* directory. You can paste this into your SQL query and modify this as needed or send this to anyone in your organisation who has access to AddressBase to execute for you.

Export the output from the query as a csv file and save it to the *data* directory of this tool.

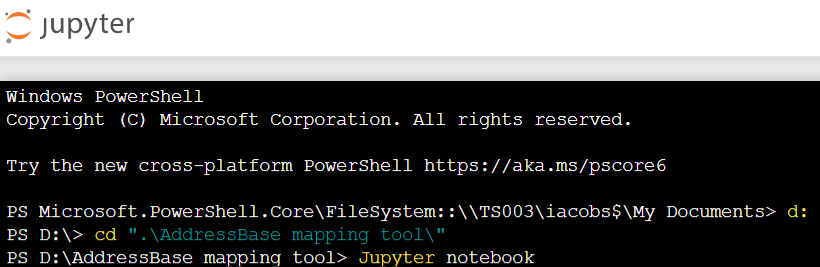
The changes to the column names using AS are important because those names are what the tool will look for. The function used for spatial table merges in the code truncates column names that are longer than 10 characters and it is easiest to manage it at this point in the process. While the class\_code column name can be overridden in the tool, it’s not recommended to do so unless necessary (and even then only to a column name of 10 characters or less).

### Jupyter Notebook

Open Jupyter Notebook and navigate to this tool’s folder. If you have saved this tool outside of Jupyter Notebook’s default directory, you will need to navigate to it using the built-in terminal. To do this, from the Notebook’s file browser select *New* and *Terminal*



This will open up a terminal window much like Command Prompt. Use this to navigate to the desired directory and then write *Jupyter notebook* followed by Enter. This will open another file browser in the directory you’ve chosen.



### First time using the tool

The tool is pre-loaded with some basic data so you can just dive right in.

All you need to do is run the cells in order. We are building a map of post boxes and post offices in England and Wales.

Cell 1 loads in the locations of the files needed, and some descriptive information. This is already filled in so run the cell.

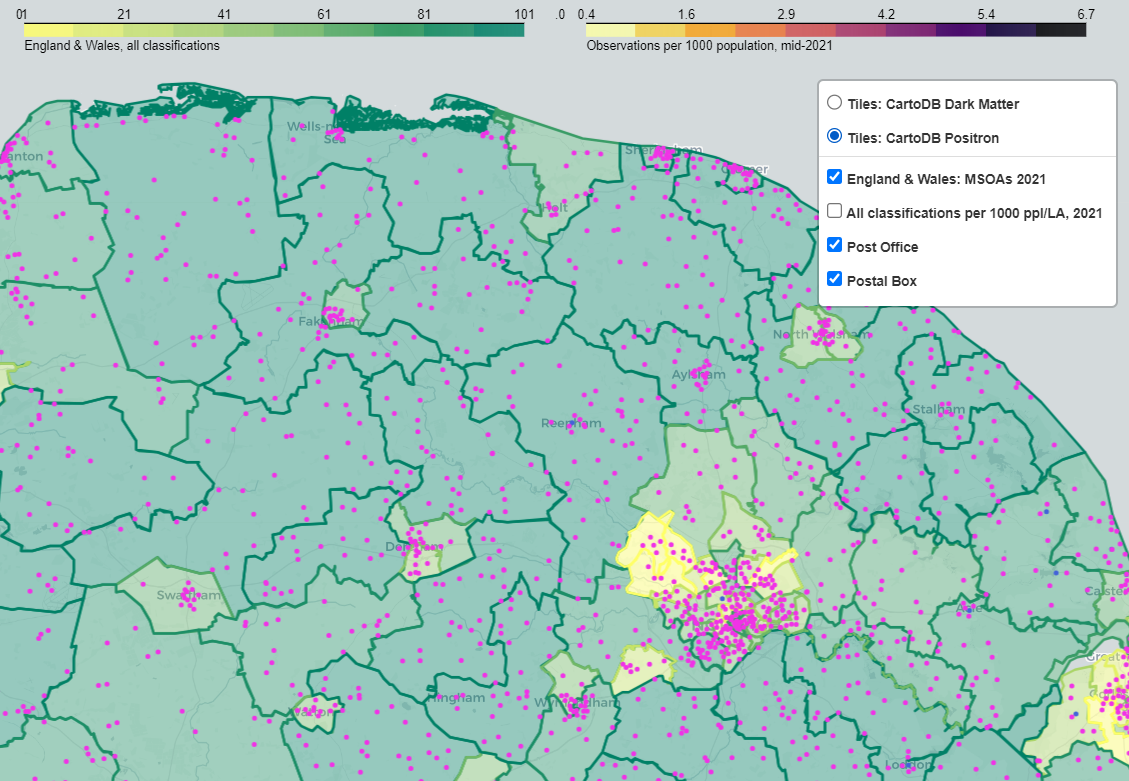
Cell 2 contains settings which affect the maps and file outputs. You don’t need to worry about these for now but they do need to be loaded in, which is done by running Cell 2.

Cell 3 contains the code which manipulates the data identified in Cell 1 and which will be used in the map. This can take a short time to run, especially if the geographies involved are complicated (lots of small areas or points). Run this next.

Cell 4 actually generates the map. Run this and a small version of the map will appear under the cell.

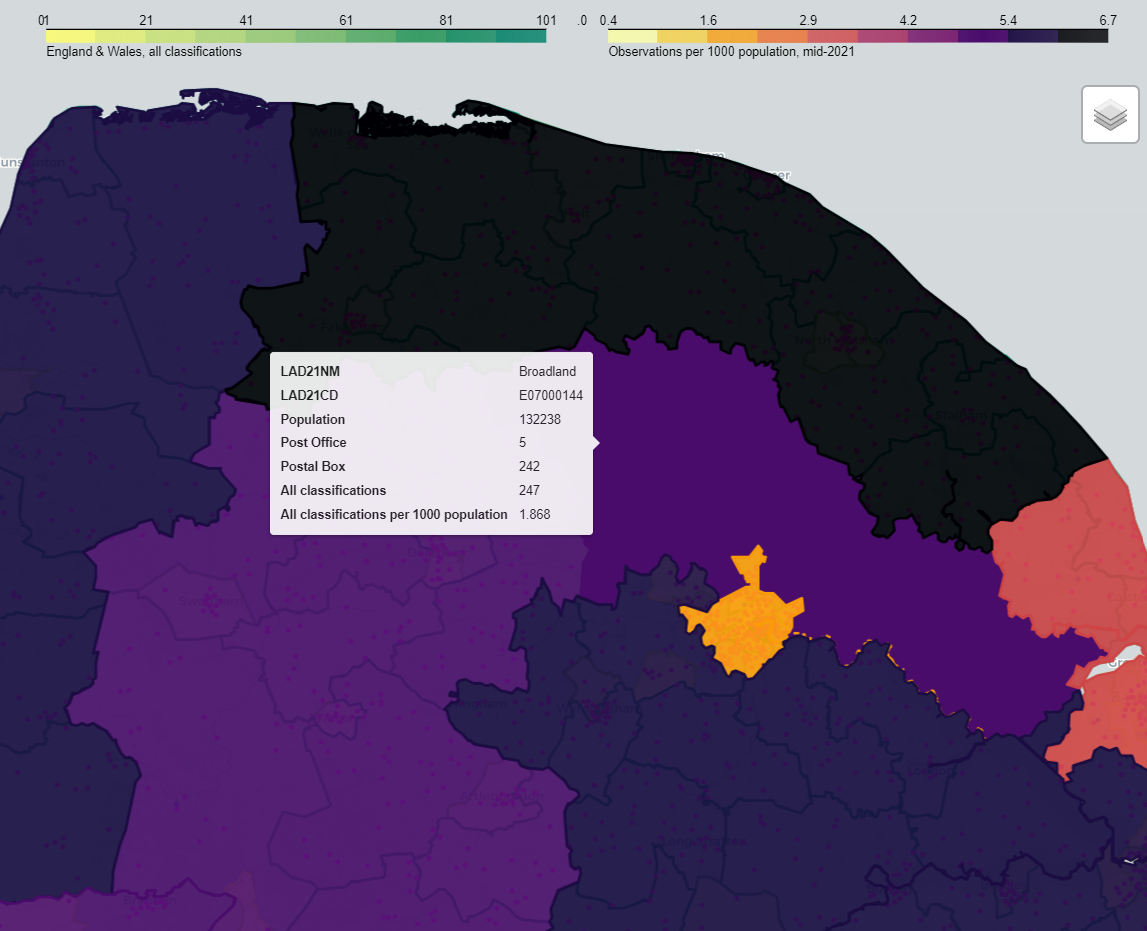
Cell 5 will open a new browser tab with the map at full size.

This view will start zoomed out, and as there are a lot of points the map will seem quite dense with them. Zooming in shows a clearer picture, so I’ve moved to Norwich to show the features of the map.



At the top of the screen are the legends, one for each loaded boundary map as well as one for the currently hidden “All classifications per 1000 population” layer. Each map has a different colour scheme to make differentiating them easy.

On the right are the layer controls. They will start off collapsed, as pictured on the left, but on hover show options that let you control what you’re seeing on the map. You can turn off points for post offices and post boxes individually, hide or show boundaries loaded into the map, and change from light mode to dark mode.

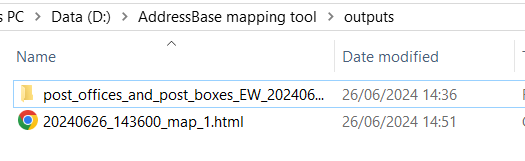


This map shows the “All classifications per 1000 population” when selected. As you can see, when hovering over any part of the map, a popup will appear to show details (this happens on all layers and also for points, not just for the population map). The last layer made visible will always appear over other map layers, so if you want to see the points overlaid on top of this you go back into the layer control and toggle them on and off again.

When you are done here, return to the notebook. It won’t respond to any other commands until you close the browser window (and might involve a bit of a wait even after that). Once the numb

Cell 6 will save the map as a html file to the outputs directory.

Having done this, have a look in the *outputs* directory. You should see something like this:



The html file is the map you saved in Cell 6. The directory contains shapefiles created when converting the SQL query’s output to a geodataframe. If you move this to the data folder and update Cell 1 with the .shp file’s location inside that directory, you can reuse it (or just delete it if you don’t want it anymore).

Once you’ve done this, try again! The *samples* directory includes a csv file called “emergency\_services\_ew\_e109.csv” showing the location of various emergency service addresses in England and Wales. Move it to the *data* directory and go up to Cell 1. Then change the *point\_data\_file* variable to:

point\_data\_file = r" emergency\_services\_ew\_e109.csv "

Then run everything again!

## Settings

### Cell 1 : File settings

*point\_data\_file* is the csv or shp file from which the point geometries will be plotted. It must be located in the data folder, however subdirectories are allowed (in the case of shp files this is better because they come with other files that must be in the same directory). The shp file is included as an option here as if the SQL output has already been turned into geographical data it saves a little bit of time and computing, so if you’re using the same data a lot this is preferable.

*point\_data\_classification\_code\_column\_name* should default to class\_name if you’ve followed this process. This is the column in the data which contains all the classification codes. If this column is called something else, change it here. This must be 10 characters or less due to restrictions on column lengths in geodataframe merges and could be something to improve in future updates.

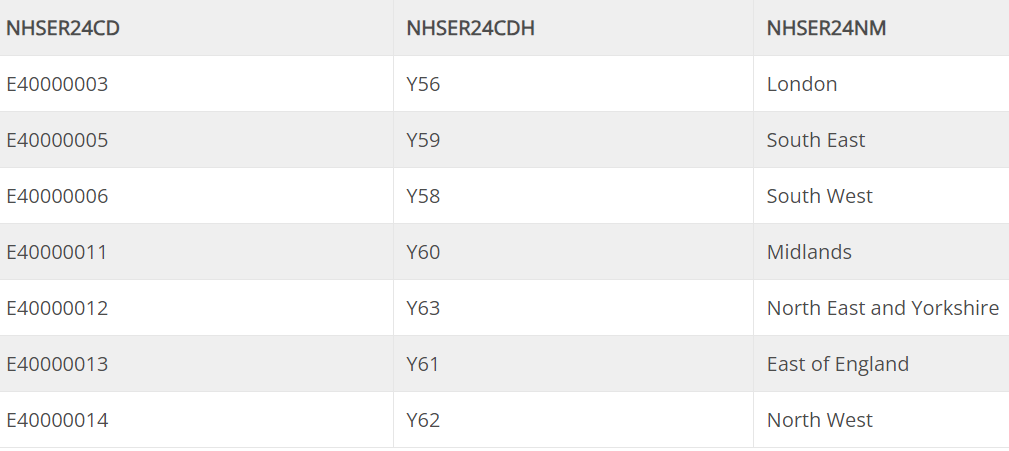
*point\_data\_classification\_description\_column\_name* should generally be left as class\_desc. If for some reason you would like it to be called something else then change it here. There is a 10 character limit on this too.

*polygon\_data\_file\_1* is the location within the data directory of the polygon geometry you are plotting your points against. It should always be a .shp file.

*polygon\_data\_file\_1\_description* is a plain text description of what the boundary file is. It will appear in the layer controls on the map.

*polygon\_data\_file\_1\_short\_description* is a short description of the above. It appears under the legend on the map and is brief for clarity, to give the user an instant idea of what they are looking at.

*polygon\_data\_file\_1\_OA\_code* this is important. Maps downloaded from Open Geography Portal have an area code column used in National Statistics Lookup tables.



This is a table of NHS England Regions, the *polygon\_data\_file\_1\_OA\_code* here would be NHSER24CD. Check this for your specific map, it will usually end with a two-digit year followed by “CD”.

*polygon\_data\_file\_1\_OA\_name\_override* can usually be left blank, as by default maps with area codes like the above will have a matching column ending with the same two-digit year followed by “NM”. If this isn’t the case for your map enter the name of this column here.

There is room for four maps in this section, in addition to the point data. These will be loaded at the same time and can even overlap so it’s possible to take in a wide area or drill down on geometries (for instance you could plot regions, local authorities, middle layer super output areas and lower layer super output areas on the same map and flick through the layers). Or alternatively you could compare best-fit geographies from different countries - for instance Local Health Board areas in Wales alongside Sub-Integrated Care Board areas in England.

### Cell 2 : Mapping/general settings

*LA\_comparison* (default True) : Generates a map of 2021 Local Authorities and calculates the number of points within those boundaries per 1000 population as of mid-2021. It then displays this as a hidden layer on the map which can be toggled on in the Layer Control. Set to False to disable this behaviour if it’s not useful. This will make the map load more quickly and reduce the file size of the exported html.

*save\_point\_shapefile* (default True) : if you use a csv file as the point data, this will convert it to a shape file and save it in the *outputs* directory. You can then use the shapefile instead of a csv, saving some computing if you’re going to be looking at this map more than once or if you might want to share that shapefile for other reasons. If you don’t want the shapefile generated or don’t think you will ever need it then set this to False.

*colour\_list* : a list of colours for the points. Each classification type captured in the data will be given its own colour in order from this list. Feel free to change any or all of these to your liking.

*column* (default ‘all’) : This is the column which is used for calculating the choropleth maps. If you have a particular classification code you’d like to focus on, you can put that code in here. The code must be present in the data. ‘all’ is a count of all classifications and should suffice for most datasets so it can be safely left alone most of the time.

*scheme* (default ‘quantiles’) : There are a number of schemes which determine how choropleth maps will display. The list is included in the tool above this option, and you can pick the one that suits your needs best. This will apply to all maps.

*k* (default 10) : The number of “bins” used in plotting the above schemes. Increase this to have more different colours on your map, decrease it for broader strokes.

*colormaps* : a list of colour maps used for the polygon geometries. The first map will take the first colourmap on this list, the second the second, and so on. A full list of these colour maps and previews can be found at <https://matplotlib.org/stable/gallery/color/colormap_reference.html>

## Future improvements

The 2021 Local Authority/Population layer could be made more accurate or even removed by using population counts by postcode centroids. This could be added to any maps loaded into the tool.

A panel dash could be built to remove the need to select a column for analysis in the settings. It currently defaults to “all” (a count of all classifications inside a geometry) but can be aimed at any other classification code in the dataset. Panel works well with Jupyter notebooks and can produce a dashboard with controls for selecting one or more columns.