PROCESSING ZOOPLA HISTORIC DATA

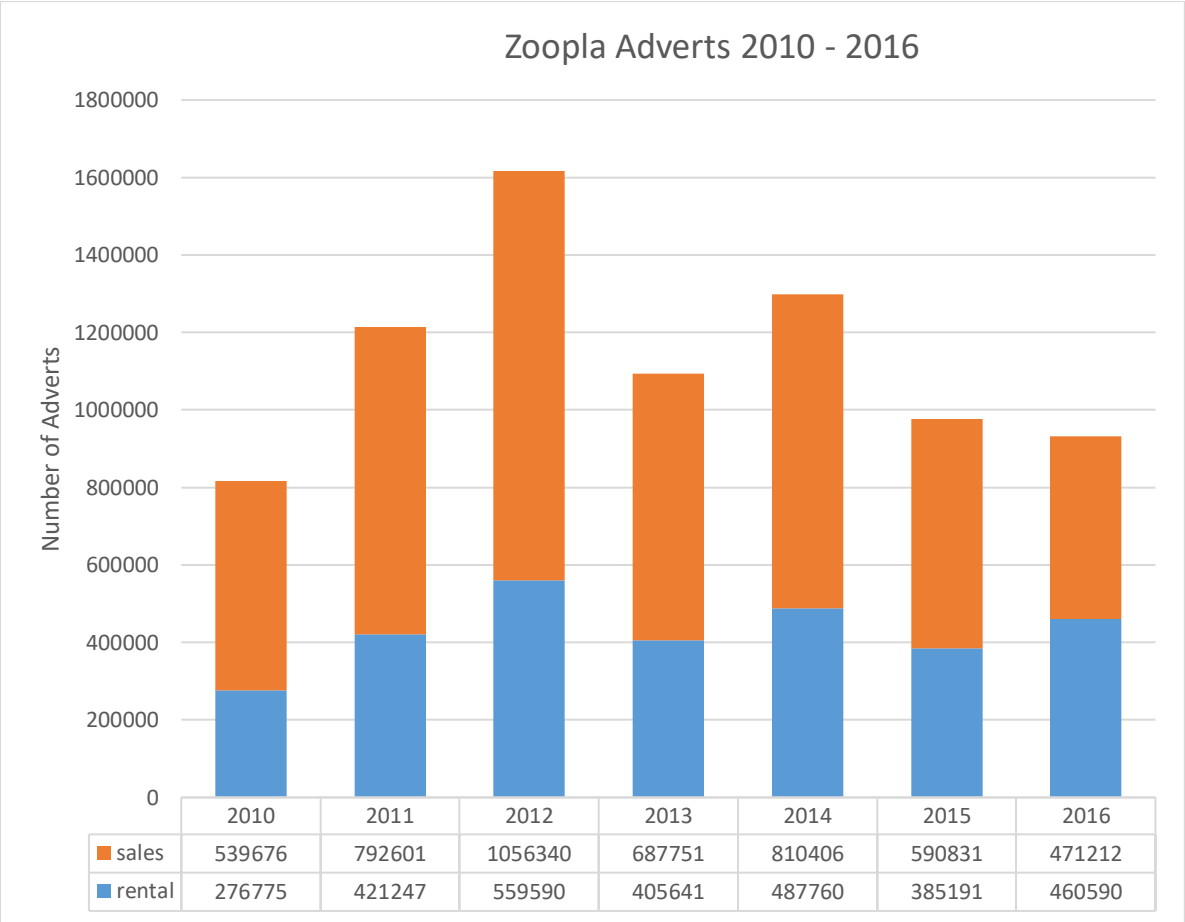
Rod Walpole

Scientific Computing Officer

Urban Big Data Centre

Zoopla has over 27 million residential property records in their archive although only a relatively small percentage of these have been advertised for sale or rental on the Zoopla website and therefore contain a property listing history. Zoopla provides access to these historic property listings via an Application Programming Interface (API). UBDC has a licence to access this API with agreement to download data for the UK as part of the Centre's housing data catalogue. This document relates to the initial task to download a baseline of the Zoopla property listing archive for Great Britain.

The process described below has yielded a historic database for GB with over 5 million records of properties advertised for sale and 3 million records of properties advertised for rent. The data run from 2010 to 2017 as full GB coverage with data available for selected areas from 2005. The number of adverts by sales or rental for 2010 – 2016 is shown in the figure below.



	2010	2011	2012	2013	2014	2015	2016
sales	539676	792601	1056340	687751	810406	590831	471212
rental	276775	421247	559590	405641	487760	385191	460590

API PROCESSING

The request to retrieve this data via the API is by individual property and requires the unique **property id** in the API request. UBDC has been collecting daily active Zoopla property listings since August 2016 (i.e. current listings) and these data include the **property id** value but as this does not cover historical data (i.e. closed listings) a method of accessing the complete Zoopla **property id** list for GB was required. The solution was to use the **Zoopla estimates** API. The requests to this API can use place names, postcode areas or user defined bounding boxes and the results returned include all property id's in that area.

The processing of the baseline **property listings history** is then comprised of the following steps:

1. Make a **Zoopla estimates** API request by area.
2. Extract individual property ids from the estimates results set.
3. Make a Zoopla **property listing history** API request for each property id extracted.

UBDC access to the Zoopla API is subject to a number of constraints:

1. There is a limit to the number of API requests that can be made per hour set at 3000 by the terms of our licence. Processing 27 million properties would take over a year to complete at this rate. This limit also applies to any UDBC API call to Zoopla services including requests made to the active listings so that if calls are made to this and the historic API concurrently they must not exceed 3000 per hour in total.
2. Testing also showed that a majority (over 75%) of property ids returned in the estimates API had no listing history but there was no way know if this was the case in advance.
3. The estimates API returns the results set by page with the maximum number of properties per page set to 100. It also limits the number of pages that it will return for an area to 99 therefore the maximum number of property ids that can be returned for an area is 9900.

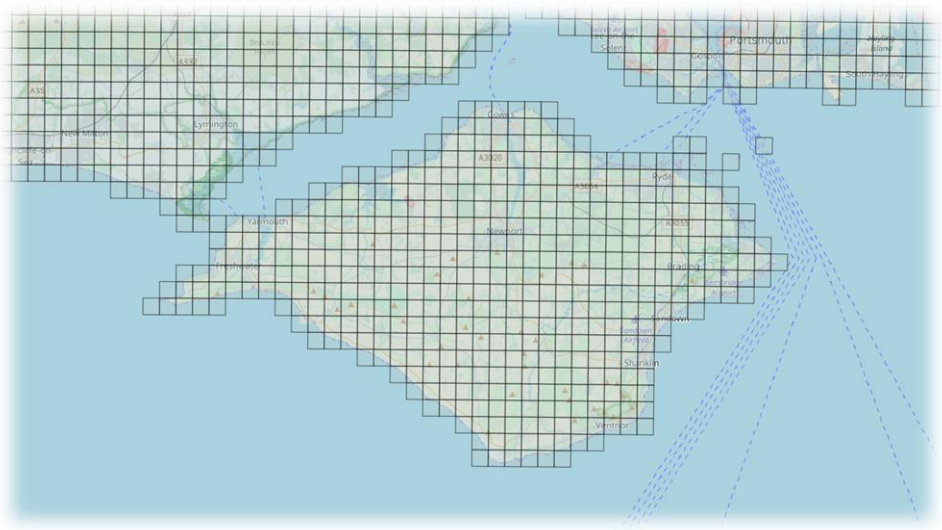
To resolve the first of these issues a temporary increase in the API limit to 7500 per hour was negotiated with Zoopla to cover the initial historic property listing processing requirement.

To resolve the second UBDC made a request to Zoopla for a change to the estimates API to add a **has historic record** flag to the results returned by the estimates API. Using this flag the results can be filtered so that only those with a listings history are used in **property listing history** API. This significantly reduces the API requests required and combined with the increased call allowance meant that the estimate to complete the task was reduced to 3 months.

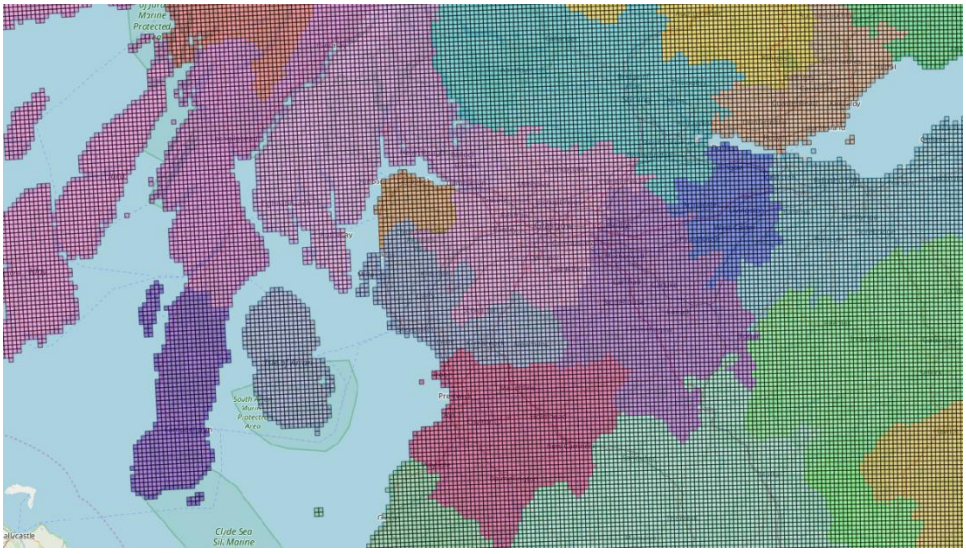
The third issue requires the use of an area boundary that ensures that the whole of GB would be processed as efficiently as possible, balancing the area request to maximise the results returned without exceeding the 9900 property id limit. It was decided to use a 1km bounding rectangle based on the Ordnance Survey's GB grid.

GRID PREPROCESSING

An example of the OS 1km grid coverage of the Isle of Wight is shown below.



UBDC has defined urban areas of interest using the 2011 census Travel to Work Area (TTWA) boundaries. The OS 1km grid squares were linked to TTWA areas using a simple majority area rule, an example of the output from this preprocessing is shown below.



The TTWA link enabled grid squares covering specific regions to be prioritised in the processing task although our aim was full GB coverage ultimately.

A final step in grid pre-processing was to convert the grid from its OSGB projected spatial reference system (EPSG:27700) to the geographic spatial reference system WGS84 (EPSG:4326) required by the Zoopla API. The final grid is stored as a PostGIS object *osgb1k_to_ttwa_wgs4326* in the PostgreSQL *housing* database (130.209.67.134). The table is described below:

Column	Type	Description
tid	Serial NOT NULL	Primary Key
geom	geometry(Polygon,4326)	grid square geometry object
plan_no	character varying(6)	OS grid square id
ttwa11cd	character varying(80)	TTWA code
ttwa11nm	character varying(80)	TTWA name
int_area	double precision	Intersection area of grid square with TTWA

ZOOPLA API PROCESSING

The processing workflow for each of the steps above was designed using SAFE Software FME product. Initially each step was run separately as described below, these are now combined into a single end-to-end workflow for efficiency.

1 – ZOOPLA ESTIMATES API

Summary: In this initial processing step the user selects the TTWA(s) to be processed. The 1km grid squares for this area are extracted from the database, an estmates API request made based on the extent of the square and property estimate results stored in the database.

1. The user updates the list of TTWAs to be selected using either the name of the TTWA (ttwa11nm) or code (ttwa11cd). The user starts FME processing workflow.
2. Matching records in the *osgb1k_to_ttwa_wgs4326* table are selected including the geometry object.
3. The minimum and maximum X and Y coordinates of the grid square geometry are extracted as _xmax, _xmin, _ymax and _ymin.
4. Zoopla estimates API request using the grid square extent coordinates e.g. https://api.zoopla.co.uk/api/v1/zoopla_estimates.json?api_key=*xxxxxx*&lat_min=*ymin*&lat_max=*ymax*&lon_min=*xmin*&lon_max=*xmax*&page_number=*[1-99]*&page_size=*100*
5. The data is returned from the API as json containing the first page of up to 100 property estimates and other processing paramaters including the result_count with the total number of Zoopla properties in the grid square.
6. The result count value is tested to determine whether it is less than 9901.
    a. If this test is passed the number of iterations of the API request is calculated.
    b. If the test fails the grid id (plan_no) and TTWA details are stored as a record in the *zoopla_estimates_errors* table.
7. When the test is passed and
    a. the result_count is less than 101 then all the data was retrieved in the first page request and this is stored in the *zoopla_estimates* table and the next grid square is processed.
    b. the result_count is greater than 100 the processing loops requesting the API for page 2 and subsequent pages until it exceeds the value calculated in 6a. The initial page 1 request and subsequent pages are stored in the *zoopla_estimates* table as described below.

Column	Type	Description
tid	Serial NOT NULL	Primary Key
geom	geometry(Polygon,4326)	grid square geometry object

plan_no	character varying(6)	OS grid square id
ttwa11cd	character varying(80)	TTWA code
ttwa11nm	character varying(80)	TTWA name
_response_body	text	json page returned by API
Zoopla_result_count	integer	Total number of properties in grid square
I	integer	Counter for page iteration
pages	integer	No. pages (calculated)
published	text	Timestamp API called (YYYYMMDDHHMM)

2. EXTRACT PROPERTY ID'S FROM JSON DOCUMENT

Summary: The estimates data extracted in the previous processing contains the property id's required in the historic_listings API. This processing step extracts **property_id** and the **has_historic_record** flag from the json data.

1. As step 1 part 1 above.
2. The json data page contains a nested array of up to 100 properties with estimate attribution. The **property_id** , **address** and **has_historic_record** attributes are extracted from the json and stored in the *zoopla_estimate_property_id* table with other parameters as described below.

Column	Type	Description
plan_no	character	OS grid square id
ttwa11cd	character	TTWA code
ttwa11nm	character	TTWA name
published	character	Timestamp API called (YYYYMMDDHHMM)
property_id	character	Unique property identifier
address	character	Property address attribute record
retrieved	date	unused
has_historic_record	character	Set to 'yes' or 'no'

3. ZOOPLA HISTORIC LISTINGS API

1. As step 1 part 1 above.
2. Records are extracted from the *zoopla_estimate_property_id* where the **has_historic_flag** attribute is set to 'yes'.
3. The system time is checked and based on this processing is throttled to allow either the maximum hourly API request rate (7500 / 3000) or, when the time is between 1:00am and 9:00am, further reduced (to 1600) while the daily processing of the live listings API is active.
4. The historic listings API is called for each **property_id**
   https://api.zoopla.co.uk/api/v1/property_historic_listings.json?api_key=*xxxxxx*&*property_id=nnnnnnnnn*
5. The json page returned by the api is stored in the *zoopla_historic_listings* table

Column	Type	Description
plan_no	character	OS grid square id
ttwa11cd	character	TTWA code
ttwa11nm	character	TTWA name
extract_date	character	Timestamp API called (YYYYMMDDHHMM)
property_id	character	Unique property identifier
listings	text	Json page returned
result_count	integer	Total number of properties in grid square
has_historic_record	character	Set to 'yes' or 'no'