DAFNI PILOT 2: UK Housing Market Models

Table of Contents

Glossary

Item	Definition
Agent based model	A model to simulate the interactions of autonomous agents, such as households, to assess their effect on the overall system (in this case the UK housing market).
Containers	Containers allow a developer to package up an application with all the parts it needs, such as libraries and other dependencies, and ship it all out as one package.
Docker	Docker is a tool designed to make it easier to create, deploy and run applications by using containers.
Docker-compose	It is a tool for defining and running multiple container Docker applications.
Kubernetes (k8s)	Kubernetes is an open-source system for automating deployment, scaling and management of containerized applications.

Key Benefits of this Pilot

- A user interface has been built that allows the UK housing models to be run simply without any need to understand the packages or how they are programmed.
- The software has been packaged so that it can be made available to remote users via the web-based interface without any need for software installation by the user.
- Increased capacity to run many jobs at once allows much greater throughput compared to running on a single laptop.
- The ability for users to inspect the many output values of the program and compare them with each other in a simple graphical interface.
- The new interface allows input parameters to be varied between runs and to compare the results to understand the effect of these values on the model.
- Simple sensitivity analysis can now be performed to see to what extent input parameters influence the model output. This is useful to test the dependence of the predictions on the input data.
- Easy visualisation of the output results has helped highlight a few issues with the simulation which were then updated. This has helped in improving confidence in the software.

Introduction to DAFNI Pilots

DAFNI will provide the National Platform to satisfy the computational needs in support of data analysis, infrastructure research and strategic thinking for the UK's long term planning and investment needs. The platform will support academic research that is aiming to provide the UK with a world-leading infrastructure system that is more: efficient, reliable, resilient and affordable. DAFNI will support big data analytics, simulation, modelling and visualisation.

DAFNI Pilots are a series of projects that run alongside the DAFNI core platform development and seek to take existing established infrastructure codes and implement them in a Cloud based environment that emulates the expected future DAFNI system. DAFNI pilot projects are submitted by the members of the DAFNI community and projects are chosen based on proposers' resource availability and benefits to DAFNI such as validating DAFNI's components, stress testing the DAFNI hardware etc. Each pilot project typically runs for 3-6 months and is supported by the DAFNI pilot team, consisting of 2-3 software developers. This will enable the following benefits to DAFNI and its community. They will:

- Demonstrate the capabilities of the DAFNI infrastructure.
- Feed the community requirements into improving and maturing the DAFNI infrastructure.
- Provide early access for the modellers to test their models on the DAFNI platform.
- Provide additional access to infrastructure models that may form part of the DAFNI service.
- Allow exploration of visualisation techniques useful to infrastructure modellers.
- Highlight typical data set requirements for infrastructure research.

Overview of UK Housing Market Models

Accurate modelling of economic systems is, in general, a difficult process, as can be seen from previous events like the 2007-8 financial crisis, which was poorly predicted by many modellers. One approach that is now being used by researchers at the Institute of New Economic Thinking in Oxford is to employ an Agent Based Model (ABM) to predict the behaviour of the UK housing market over time and how this is influenced by external factors such as the trend to invest in the Buy-To-Let (BTL) market.

ABMs simulate a system of agents, or actors, over time as they follow objectives designed to represent real systems. In the case of the UK Housing market model the prime agents represent households, who have incomes, pay taxes and other bills, while needing to find housing either by renting or buying property. Other agents in the model include banks that will lend to the households, and a central bank that sets policy and interest rates. By adjusting the various input parameters to the system and running the simulation over a sufficiently long period of time the model should be able to predict the peaks and troughs in the housing market. Having calibrated the model to mimic the known history of the housing market it should then be possible to simulate the effect of policy changes on the market. This will enable economists to predict how policy changes will influence the market. The details of an earlier version of the model used here are given in a Bank of England working paper (Baptista, et al., 2016).

There are two versions of the model at present:

- The non-spatial model, which just has a single market to represent the overall UK housing market. This is the model that was available at the start of the pilot project and gives an integrated view of the whole system. However it is unable to represent regional variations which are known to be significant.

- The spatial version of the model allows for a number of distinct housing regions along with commute costs and travel times between regions. This will be a much more computationally demanding model, particularly when trying to represent many regions. This model was not ready when the pilot was started and the version that is now available, while able to model multiple regions, is still in a state of development.

Most work of the pilot project has focused on the non-spatial version as this was available and better tested. An initial version for the spatial model has now been developed but has not been extensively tested.

Analysis of Software

Non-spatial Housing Model

The model is written in Java and uses the Maven build tool to manage the software from a github repository. Approximately 6000 lines of code are contained in 38 source files. The simulation reads a `config.properties` file that defines 89 parameters which control the model. Many of these are derived from other studies, analysis of market data and "tuning" of the model. As well as these parameters there are a few small CSV files that contain fixed data tables such as for the tax and national insurance rates. Note that though the simulation can run over many years, the system does not include inflation or changing tax rates, etc. Instead it represents the evolution of the system in a steady state.

Some of the parameters have a direct effect on the simulation run and memory requirements. These include:

- TARGET_POPULATION – this is the number of agents to be used to represent the households in the model. This is much less than the actual number of households in the country, since each agent can act as a proxy for a large number of similar households. The default is 10000 to represent the circa 26 million actual households in the UK. The run time of the model increases with this parameter at a rate slightly above linear. The memory requirements will increase linearly with this parameter.
- N_STEPS – the number of time steps to run the model for which defaults to 6000. This represents the number of months to run the simulation for, and the run time increases approximately linearly with this value. A period at the start of each run (circa 2000 steps) is required for the simulation to reach "equilibrium" and then various statistical measures can be made on the remainder of the run. Hence this parameter may be increased to reduce the uncertainty in the measured values, though run time increases linearly with this value.
- N_SIM – this number of simulations to run. The default is a single simulation, but if this value is set to 2 or more then multiple runs will be made, each of the same system but using a different random seed to initiate the run. This gives information on the variance in values measured from the simulation due to its random nature.

The code is single threaded at present so does not make use of multicore processing, and typical run time with the default parameters is about 60 seconds. The memory requirements of the model are also modest using the default parameters and can easily run in the default Java VM size.

Outputs of the Non-Spatial Housing Model

The main output of the model is a CSV file that details the monthly values of a wide range of measures of the simulation. These include measures such as the house price index (HPI) for sale prices and a similar measure for rental properties. In total there are 48 output values provided.

These can be displayed as time graphs of the value which often shows a semi-periodic response that reflects rises and falls in the housing market. While it is possible to compare these measures as input parameters are varied, it is difficult to make quantifiable measures from the graphs themselves. To extract simple measures of the parameters an additional processing step was added to extract means and standard deviation of these parameters. The standard deviation in particular is a measure of the variation of a value over the time period and may show if varying an input parameter influences the amplitude of fluctuations in the sale HPI. However the standard deviation gives no indication of changes in the frequency of fluctuations. To measure this an addition was made to compute the frequency response of the sale HPI. This was done using R and the spectral density estimation function “spectrum”. The additional tables are generated as CSV files after the main Java program runs and all results can be stored into a database.

Figure 1 shows the typical response of the model in terms of the sale and rental HPI after the model has reached equilibrium.

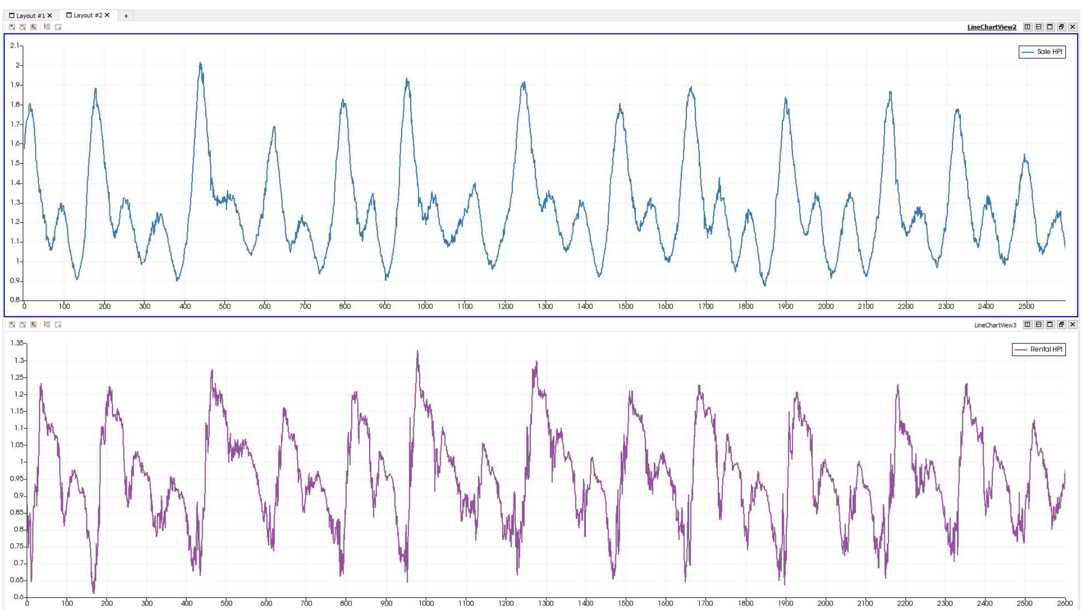

Figure 1: Time response of the Sale and Rental HPI of the non-spatial housing model

Figure 2 shows the estimated spectral response of the model for the sale HPI.

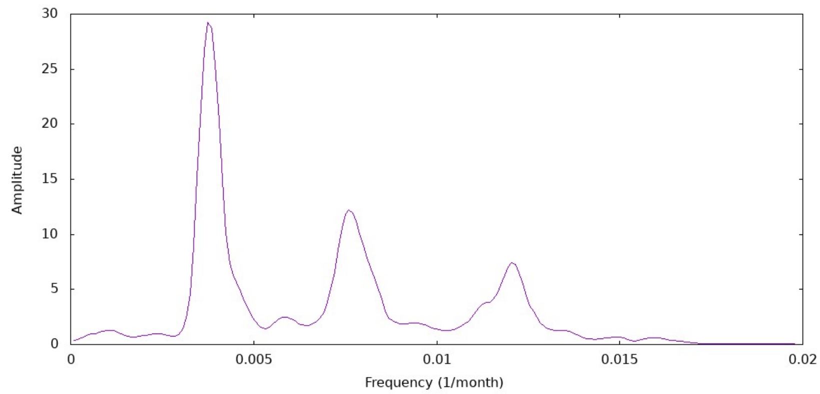

Figure 2: Spectral response of the Sale HPI over a simulation, calculated using R.

Spatial Housing Model

The spatial housing model is an extension of the non-spatial model that allows for a number of separate regions. Each region is run as a market in its own right while travel and cost measures between regions represent the average cost of commuting between them. An agent household may choose to live in a different region to that of their place of work, but has to include the travel costs (both time and money) in their decision. This model is still in an early stage of development and the test case uses three regions with matrices that represent the travel and time costs. These are defined in a small set of CSV files that are included in the Java resources directory. These can be replaced to represent any number of regions as long as suitable time and cost metrics are provided. Figure 3 shows how the three region case can be represented. At present the model does not include physical distances and directions between regions so these can only be approximately implied from the travel times. In the final model it may be that physical coordinates of the regions are included, which would allow display of data on a real map display, of the type used in the Pilot 1 study for digital communications.

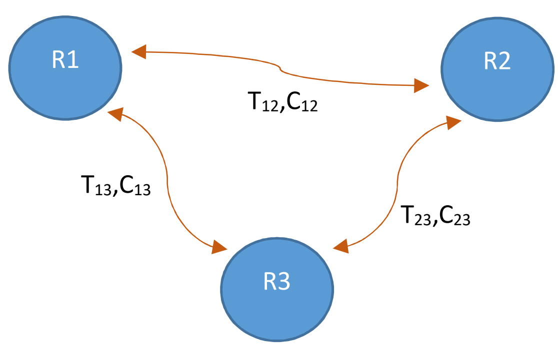

Figure 3: Three region spatial model with time (T_{nm}) and cost (C_{nm}) metrics for travel between regions. Regions could be local authority areas. The model also includes fixed times and costs for commute within each region.

The goal of the spatial model is to understand how house prices are influenced by travel time and cost. This will be useful in understanding the effect of building new travel links and the pricing policy used for them.

Outputs of the Spatial Housing Model

The spatial model provides a very similar summary CSV table to that of the non-spatial model for the behaviour of the total market. This data can be treated in the same way as before, with post-processing to get the average and standard deviation values along with the spectral response of the overall sale HPI. To see regional variations, an additional CSV file is generated for each region in the model, with a similar set of time dependent values. The means and averages of these values can also be extracted. These values can be displayed using the same techniques used for the non-spatial case.

For regional results an additional option is added to the GUI to allow the user to select between display of the regional results or the overall system values. This allows comparison of the regional variations. Figure 4 shows an example of the variation of measures over time for the 3 region case.

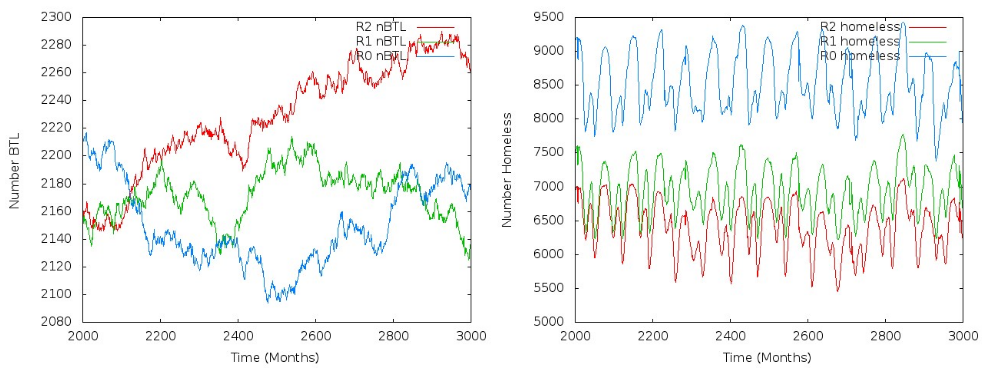

Figure 4: Regional data compared for the 3 region model for the number of BTL properties and the number of homeless households. Homeless in this context may mean leaving with parents while looking to buy or rent.

The ability to view the model outputs quantities in a simple way through the GUI is helpful in debugging and understanding the model. For example an issue which resulted in negative bids for rental properties, which occurred under certain circumstances, was identified and corrected by the software author.

Challenges

The non-spatial model is well developed and has been used to make predictions on the effect of the BTL market on the overall stability of the UK housing market. However it does depend on a large number of input parameters which have been derived from various sources. It was also provided as a Java Maven application where the input parameters are contained in a configuration file and the results are just returned as a large CSV file. Many of the potential users of this package are not computer programmers and would have difficulty in running this package directly and displaying the results in a meaningful way. The main aims of the pilot for this application were to:

- Provide a simple way to run the application on a cloud based infrastructure such as that to form part of DAFNI.
- Run several instances concurrently to investigate parameter dependence.
- Allow the user to vary the input parameters through a simple graphical interface that does not depend on knowledge of the programming involved.
- Give a simple way to view the time dependent results of the simulation as 2D graphs and to compare results with different parameters.
- Extract measures from the data to allow direct comparison between runs. These include the standard deviation of variables and the spectral response of the Sale HPI.

While the Java software itself can run on most systems and give consistent results, it is useful to bundle the package as a Docker instance. This avoids the need to install the supporting packages (Maven and Java) on systems and allows easy deployment on Kubernetes.

The spatial model presents many of the same issues as the non-spatial case, though it is in an earlier stage of development. The ability to quickly inspect the results of the simulations should help in identifying issues with software as its functionality is extended.

While the spatial model currently only deals with a small set of regions the run time is comparable with that of the non-spatial model. When the system is scaled up to hundreds of regions then the computation time will become much more of an issue. While DAFNI will still be able to run such jobs in a batch mode, the single threaded nature of the model will be a limiting issue. Making the code multithreaded, or MPI parallel, is not required at present but may be required for future large scale spatial simulations.

At present the spatial model does not include geographical details of the regions, just the cost of commuting between them. In future it would be useful to have the physical location of regions and to be able to display these to users. In addition it might be useful to have a way to update this map, e.g. with a new rail link which could automatically update travel times and costs.

Outcomes

This Pilot built on the infrastructure that was developed as part of the first Pilot (digital communications) using a Software as a Service (SaaS) implementation to give access to the two housing model codes via a simple user interface. This interface exposes the model parameters to the user and allows one or more jobs to be submitted to be processed and the results made available in a user interface.

The fact that the housing models depend on a number of user defined parameters that control the simulation means that a more detailed sensitivity analysis can be made. This has been implemented is a simple manner, allowing one parameter to be varied at a time.

For the two housing models this pilot has provided:

- A web interface built in the Vue.js JavaScript framework. This interface allows remote access to the models without any need to install software on the local computer. All the user settable parameters are exposed through this interface and can be altered before a simulation is submitted. The server side code runs on a machine within DAFNI and allows computational jobs to be submitted onto DAFNI cloud resources.
- A database that has been set up to store the job parameters and results from simulation jobs. This allows a series of jobs to be submitted by the user and then compared to see the effects of the changes to parameters. This allows a persistent store of simulations.
- An extended set of APIs, based on those developed for Pilot One, for submitting jobs to the DAFNI cloud infrastructure. The APIs have been extended to allow them to work with the requirements of the housing models. The database had to be updated to store the new parameters and results and the job queue adapted to allow different images to run as required.
- Appropriate visualisation tools. The visualisation requirements of the housing models are relatively simple and mainly require 2D plots of values against time. The ability to view these easily within the interface should be helpful in analysis of results. In addition to the comparison of these time graphs, it was thought useful to provide some simple analysis that

would give summary data on simulations. This was done using an R script to post-process the CSV results and give mean and standard deviation data. In addition, a simple spectral analysis is performed on the Sale HPI data, which should measure the frequency response of the system.

- Efficient sensitivity analysis. The user can submit a number of jobs that are run in parallel on the cloud infrastructure. This allows faster analysis than would be possible with a sequential approach.

A typical view of the GUI to submit jobs is shown in Figure 5. This is for the non-spatial housing model and shows the Home page of the interface. This lists the jobs that have already been run and which of these have been selected for visualisation. The user is able to select scenarios and submit jobs to simulate these conditions over the required time period. Job progress automatically updates to show the state of jobs submitted to the queue. Finished jobs can then be selected and visualised.

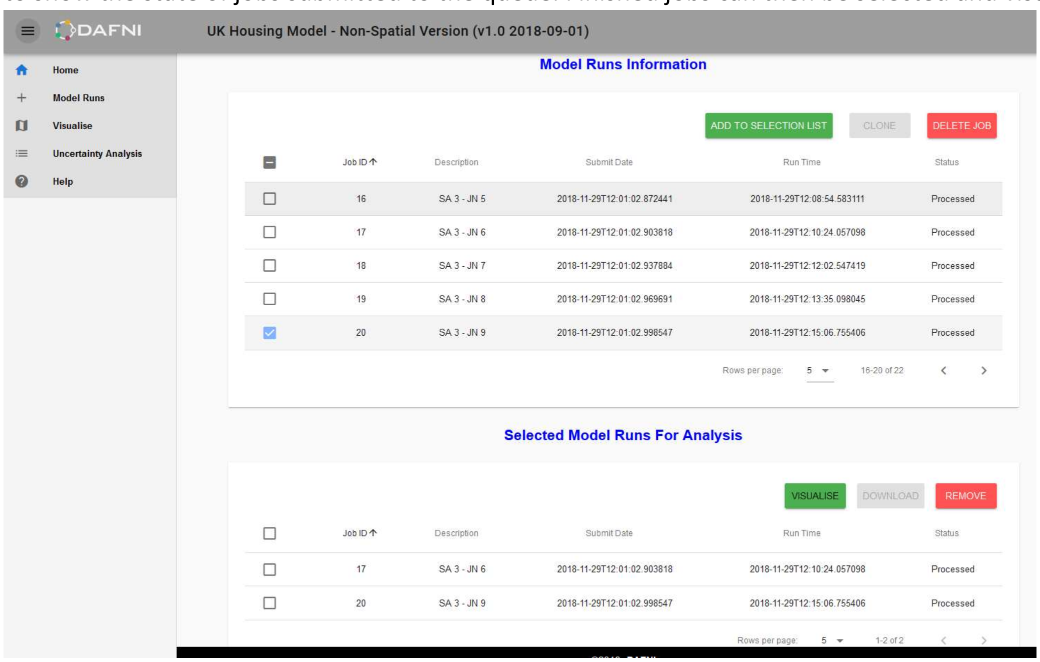

Figure 5: The interface for the non-spatial housing model home page showing runs and their statuses.

An example of visualisation for time series data is shown in Figure 6. Any of the approximately 50 output variables can be displayed and compared between runs.

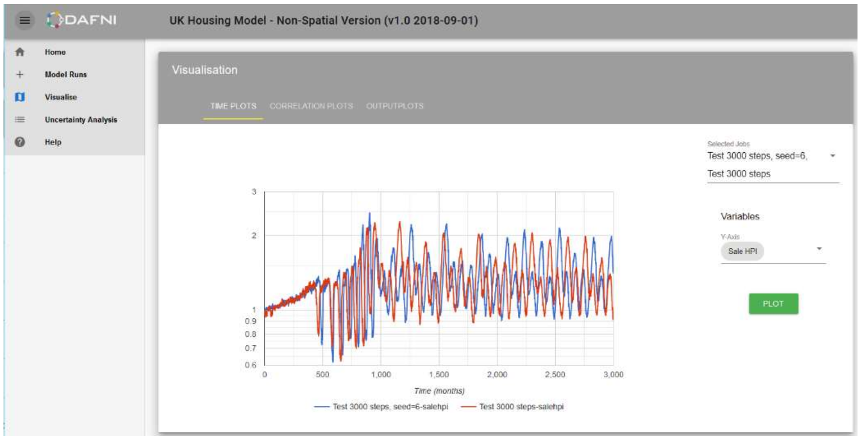

Figure 6: A time plot comparing Sale HPI between two different simulations.

Other tabs allow different visualisation options. The correlation tab plots one variable against another. An example of this is shown in Figure7. The scatter plot can help identify if the quantities are related to each other.

For the case of sensitivity analysis, output variables can be displayed as a function of the input variable that is varied to see its influence. The user can select an input parameter to test and define the range and number of simulations to run. These jobs are then submitted to the queue and run with one click. When the results are ready they can be inspected using visualisation option as shown in Figure 8.

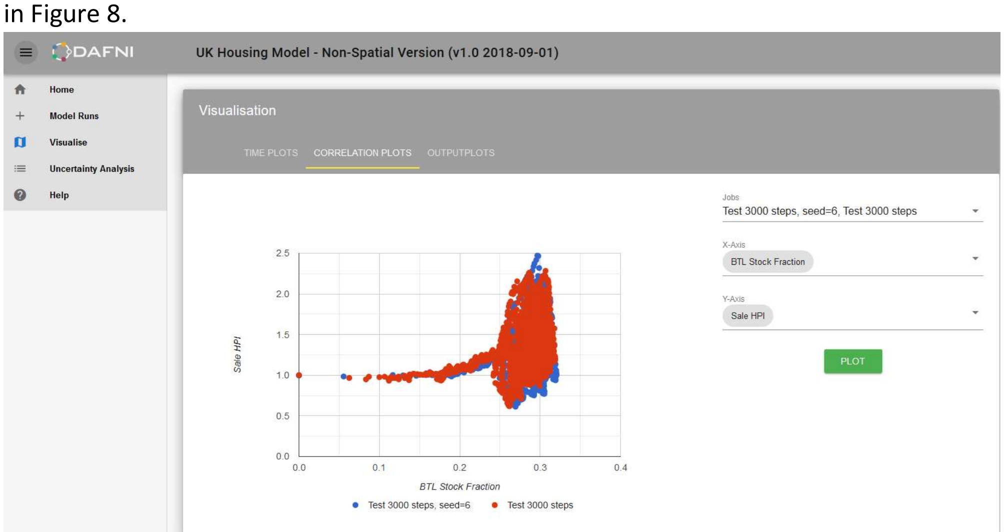

Figure 7: Correlation plot example of BTL fraction vs Sale HPI.

For the spatial model there is the option of displaying variables for each of the individual regions as well as the summary versions of the data. When the spatial model is extended to include a larger number of regions, it would be useful to add map based visualisation of results as used in Pilot One.

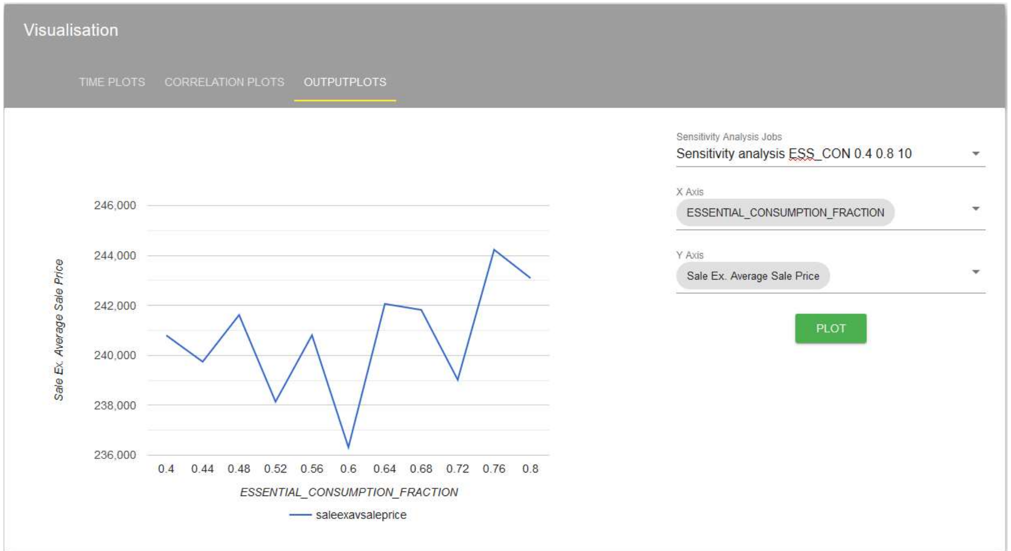

Figure 8: Sensitivity analysis plot. The Average sale price against the essential consumption fraction parameter.

Benefits

The main benefits derived from Pilot Two are as follows:

- Java simulations, such as these agent based models, can be easily adapted to run in the job queue environment developed as part of Pilot One.
- The visualisation requirements of this project were lower in terms of volume, but required more processing to produce meaningful results, such as the spectral response. The use of R to extract this data was shown to fit well with the containerised environment.
- While default jobs run quite quickly, in the order of 1 minute, for both sensitivity analysis and spatial runs with many regions the run time can become large. The ability to submit many jobs to a processing queue and have them run in parallel will be helpful for future studies. The user can simply reattach to the user interface to inspect the progress of long running jobs.
- The provision of a simple to use interface to the software will help make it accessible to non-technical users who wish to investigate how the output of the model depends on input parameters.
- The inclusion of the housing models into the Kubernetes infrastructure helped to define how to deal with multiple jobs in this system.
- A simple sensitivity analysis has been implemented and shown to be useful in determining the dependency to the model results on the input parameters.

Lessons Learned

For the Platform

- This pilot was able to use the same job submission queue and style of web user interface that was developed for Pilot One. This showed that this framework was general enough to be applied to other modelling cases. This gives more confidence that the conclusions drawn from Pilot One are applicable to the platform development.

- Running the different applications from the first two Pilots together in the same job execution framework highlighted the need to be able start workers using the appropriate Docker image. Similarly, the Platform will need to be able to instantiate separate worker images for each application.
- Sensitivity analysis is likely to be an important requirement for many models that run on the Platform. This pilot highlighted how a simple approach to this can be implemented. A series of jobs can generated and processed in parallel to determine output dependencies. The statistical nature of this agent based model highlighted the need to consider this in sensitivity analysis. This highlights some of the considerations needed in the platform.

For the Pilots

- The software and data for the non-spatial pilot was available at the start of this project and we were able to begin the analysis of this immediately. However the spatial model only become available after the project was under way and this required some work to run alongside the non-spatial model. When the spatial model is more fully developed it is likely to offer more opportunities for map based visualisation of the results, but this was not possible in this pilot.
- Sharing the software from Pilot One with this pilot substantially reduced the total amount of work needed to develop this interface. This highlighted the need to look for generic solutions whenever possible.
- The set of parameters for both input and output for the spatial model changed during the course of this work. This highlighted the need for the pilots to track the version of the model they are built against. The pilots also need to be easy to adapt to these changes in parameters.

References

Baptista, R., Farmer, J. D., Hinterschweiger, M., Low, K., Tang, D., & Uluc, A. (2016, October). *Macroprudential policy in an agent-based model of the UK housing market.* Retrieved from https://www.bankofengland.co.uk/working-paper/2016/macroprudential-policy-in-an-agent-based-model-of-the-uk-housing-market