**Introduction**

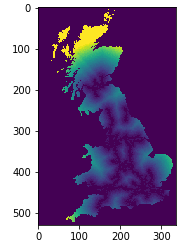
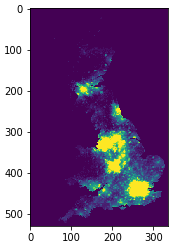
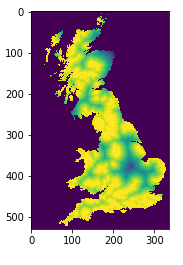
Most current GIS techniques have limitations in modeling changes in the landscape overtime, but the integration of CA and GIS has demonstrated considerable potential. The limitations of contemporary GIS include, its poor ability to handle dynamic spatial models, poor handling of the temporal dimensions. In coupling GIS with CA, CA can serve as an analytical engine to provide flexible framework for the programming and running of dynamic spatial models.

**Cellular Automata – Site Selection**

Cellular automata are mathematical idealizations of physical systems in which space and time are discrete, and physical quantities take on a finite set of discrete values. A cellular automaton consists of a regular uniform lattice (or ''array''), usually infinite in extent, with a discrete variable at each site (''cell''). The state of a cellular automaton is completely specified by the values of the variables at each site. A cellular automaton evolves in discrete time steps, with the value of the variable at one site being affected by the values of variables at sites in its ''neighborhood'' on the previous time step. The neighborhood of a site is typically taken to be the site itself and all immediately adjacent sites. The variables at each site are updated simultaneously (''synchronously''), based on the values of the variables in their neighborhood at the preceding time step, and according to a definite set of ''local rules.'' In this cases a cellular model was used to simulate a site suitability analysis process for identify possible suitable location for constructing a factory that makes rock aggregate somewhere in the UK.

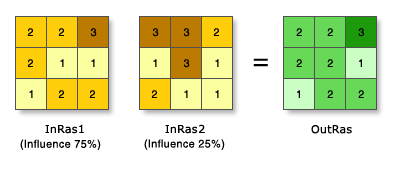
Suitability Analysis is a process that ranks and scores sites based on multiple weighted criteria. The suitability of a candidate site can be ranked based on data variables, site attributes, or proximity to point features. Selected criteria are assigned weights to determine scores for each potential site. An interactive review illustrates the final score ranks from most suitable to least suitable.

**Implementation of the model**

**Selecting the criteria’s -** To create a cellular automaton model for suitable sites, three criteria’s was chosen namely **Geology**, **Transport** and **Population**.

**Layers normalization** -before combining the three layers, it’s important to ensure that these datasets are in the same scale e.g. 0 to 9 or 0 to 255.

**weighted overlay -** the three layers above were combine using weighted overlay process. The weighted overlay is a standard GIS analysis technique often used for solving multi criteria problems such as generating surfaces representing site-suitability. This technique is used when a number of factors of varying importance should be considered to arrive at a final decision. Below is an illustration of how the process works.



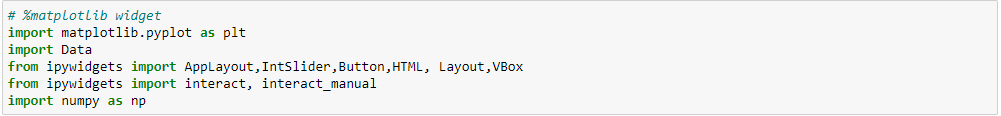
In the illustration, the two input raster’s have been reclassified to a common measurement scale of 1 to 3. Each raster is assigned a percentage influence. The cell values are multiplied by their percentage influence, and the results are added together to create the output raster. For example, consider the upper left cell. The values for the two inputs become (2 \* 0.75) = 1.5 and (3 \* 0.25) = 0.75. The sum of 1.5 and 0.75 is 2.25. Because the output raster from Weighted Overlay is integer, the final value is rounded to 2.

**How the Model Works**

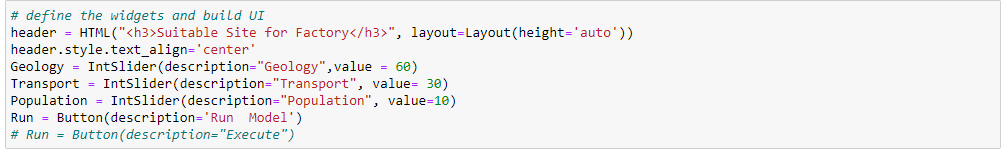
The site selection model was implemented in Jupiter notebook and in consists of

1. Input files (Geology, transport and population) – this are text files which represent the three criteria’s used to model suitability analysis;
2. Class (Data) – this python class provides a mechanism for read external data into the model; and
3. Notebook file (GEOG5003\_Assignment\_two) – this file contains the model. implementation and how it instantiates the data class to read the data, build an interactive graphical user interface
4. **Setting up the model**

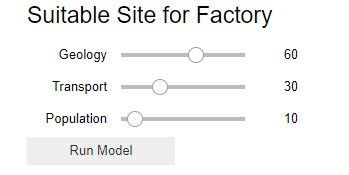
* Import the required libraries



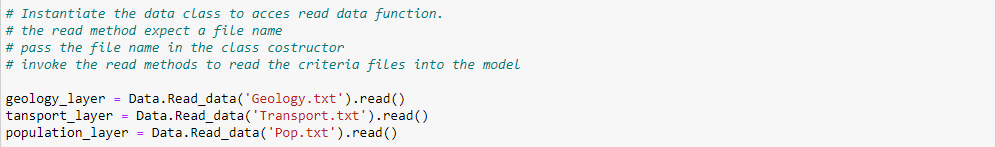
* Set up the user interface widgets;



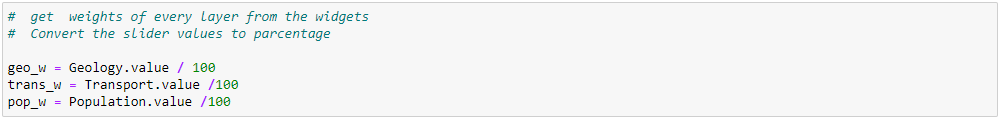
* + **Header:** title of the application widget
  + **Sliders:** (geology, transport and population)
  + **Button –** executes the model after changing the weights.



* Instantiate the data class to read in the criteria’s



* Initialize the weights from the slider widgets

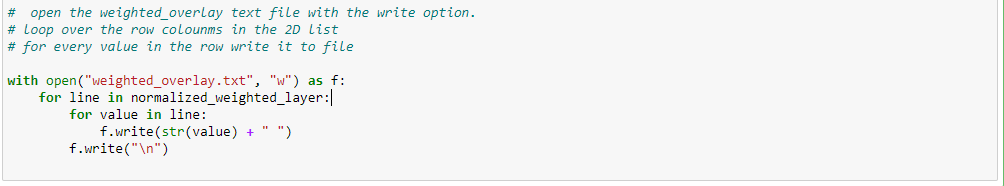


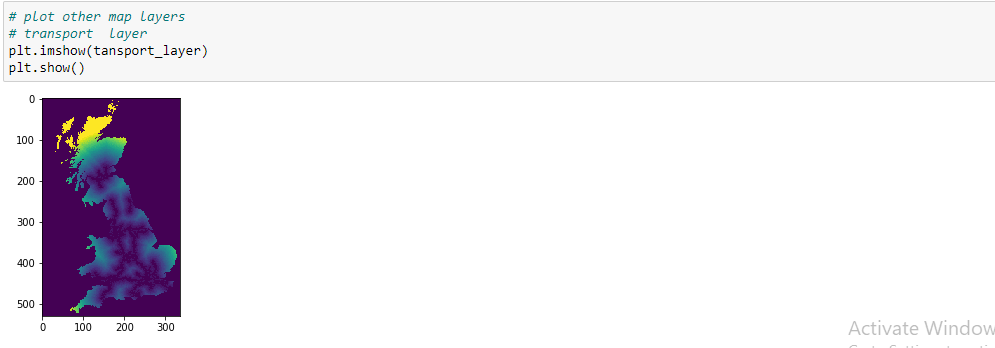
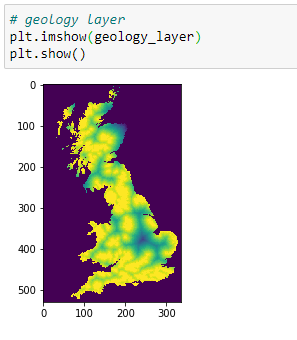
* Initialize the normalize method, this method will be used to rescale the weighted layer
* Initialize the compute\_weighted\_overlay method, this method combines the criteria layers and returns a weighted layer lists.
* Call the above method and assign it to weighted\_overlay\_layer ... this will store a list of weighted pixels for further processing.



1. **Running the model**

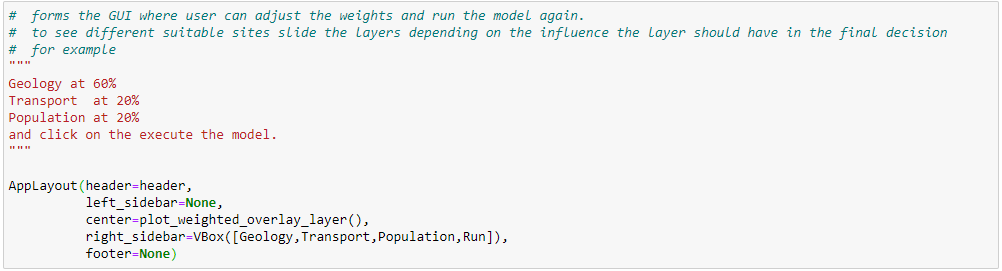
* Once you have the model up, you can export the normalized values to txt file by invoking the normalized layer method in the write command.



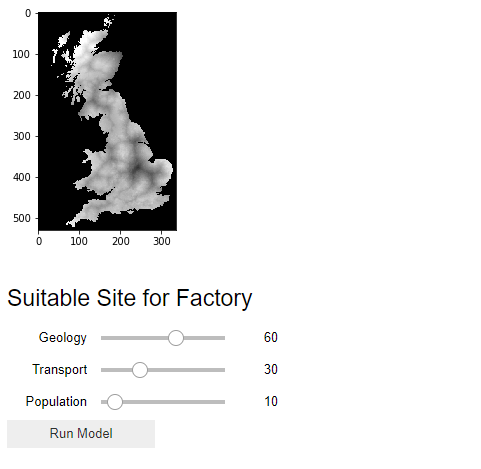
* To plat maps, pass the weighted overlay list in matplolib plot method as below.

1. **Interacting with the model**

* Run the applayout cell: app layout is ipywidget template that specifies the layout of the application. Here, we pass the GUI elements into the applayout widget and arrange them in terms pf header, side panel as shown below.



* Use the sliders to change the weights and click the run button.



* To save, map as JPEG, right click on the image and save as alternatively you can use the matplot save option.