UNIGIS Spatial Simulation Course: Assignments 1-5

Hyesop Shin

Table of contents

1 Assignment 1: Schelling Model

1.1 The Agent-based Simulation of Urban Segregation in Cambridge

Inequality has long been issued in our modern cities as the distribution of wealth within the city is unequal (link). The income, education, as well as ethnicity are a few of the the determinants that generates the imbalance in the society. This widening gap between the rich and the poor triggers the area where people want to dwell.

For example, Cambridge UK has changed significantly since 2010 by its economic wealth, new buildings invested near the Cavendish lab in the West to the immense Biomedical Campus in the south, and the urban sprawl of new towns that attract more people to enjoy the gist of the academic rigour and educational motivation. While Cambridge is known as one of the the safest and most liveable cities in the UK, there also exist deprived areas such as King's Hedges, North Chesterton, and Barnwell which are closer to each other.

The aim of this short piece is to investigate to experiment how heterogeneous agents start with no background who their neighbours are but continuously relocate their locations until they find a perfect match. It is envisaged that urban segregation is not happened from the city creation, rather happened over time.

1.2 Data Import and Setting Up an ABM

1.2.1 Query to Retrieve the map of Cambridge

The initial job is to visit the openstreepmap website to download the study area. This article visited https://overpass-turbo.eu/ and typed the code accordingly.

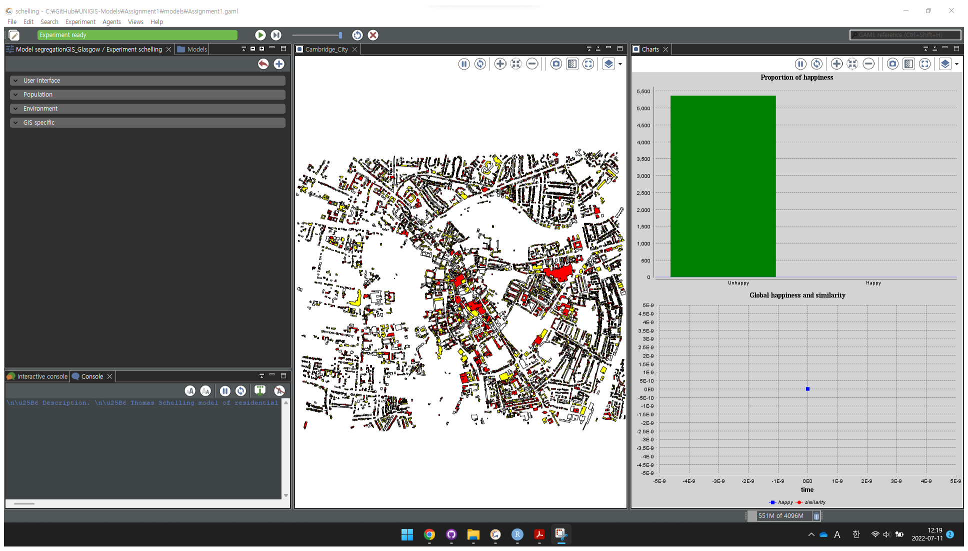
```
[out:json][timeout:50];(
way["building"]({{bbox}});
relation["building"]["type"="multipolygon"]({{bbox}});
);out;>;out qt;
```

The `geojson` file was opened in QGIS and then exported as a `cambridge.shp` file.

1.2.2 Opening GIS GAMA

In the segregation model, there are two types of buildings coloured red and yellow. The idea of segregation is that the agents (here buildings) will self-segregate over time into clusters until the happpiness reaches 100%.

Figure 1 is a snapshot of the GAMA interface waiting to get started. In the interface tab (middle), The chart on the top right shows the proportion of happiness which is a dichotomous in unhappy and happy state. The bottom right chart is a line graph that shows the happiness and similarity over time.



```
\n\u25B6 Description. \n\u25B6 Thomas Schelling model of residential
```

Figure 1: Interface of Cambridge using GAMA

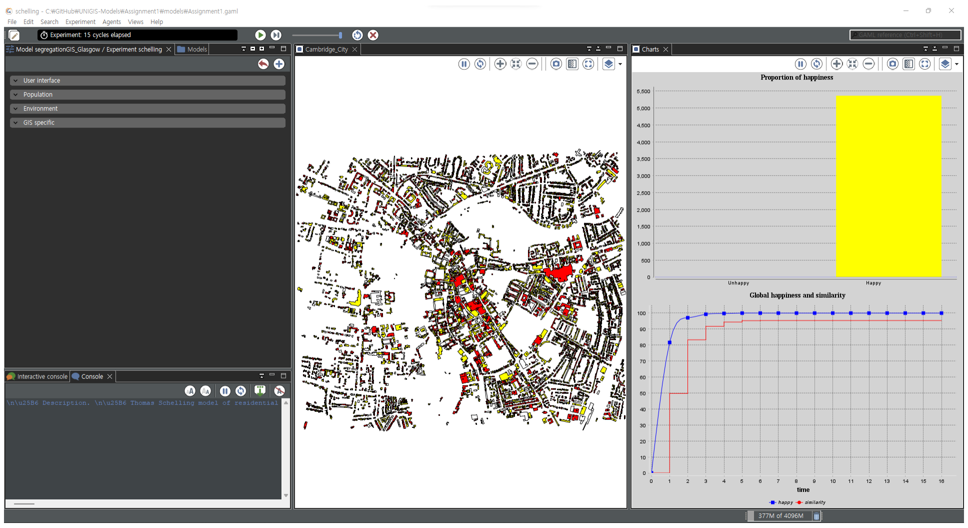
1.3 Results

Figure 2 shows that the happiness reaches 100% after the fourth cycle. This means that each agent (building) has at least 50% of their surroundings with the same colour. The similarity did not reach 100% as their are voids (parks) which exceeded the agents' range of mobility.

2 Assignment 2: Cellular Automata – hydrological run-off model

2.1 Maps

Cambridge is a city in the south-eastern part of the United Kingdom. The terrain is mostly flat and dry. As a result, flooding in and around the city is less likely. However, for the purposes of an experiment, this is a simulated result of what would happen if a waterfall affected the city (see Figure 10). As can be seen, the highest area is a cell in the bottom right corner, so the water fall would always begin there regardless of iterations.



```
\n\u25B6 Description. \n\u25B6 Thomas Schelling model of residential
```

Figure 2: Happiness

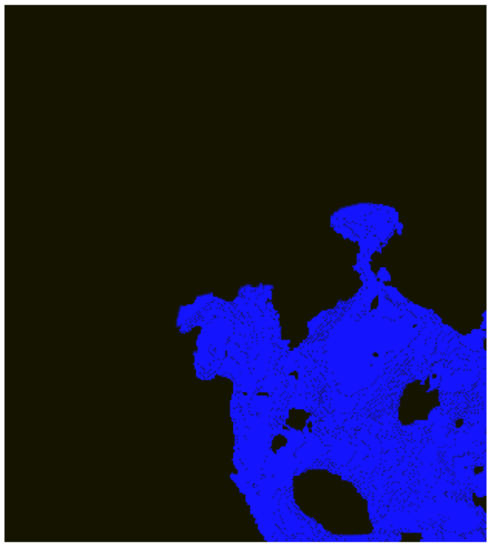


Figure 3: Cambridge Map of the Run Off after 1000 time steps

2.2 Graphs

Figure 4 depicts the water's lowest elevation over time. The starting altitude is 73 metres above sea level, but as the water flows, the lowest altitude drops dramatically to 20 metres around the 300th time step, 10 metres at the 500th time step, and finally 6-7 metres after the 600th time step.

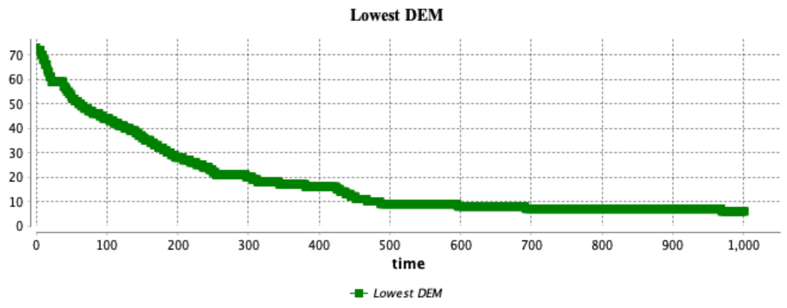


Figure 4: Lowest DEM with water

There are approximately 25,000 cells that contain water after 1000 time steps. As this map comprises 108,576 cells (312 x 348), it turns out that just over 23% of the spatial extent were affected by water run off.

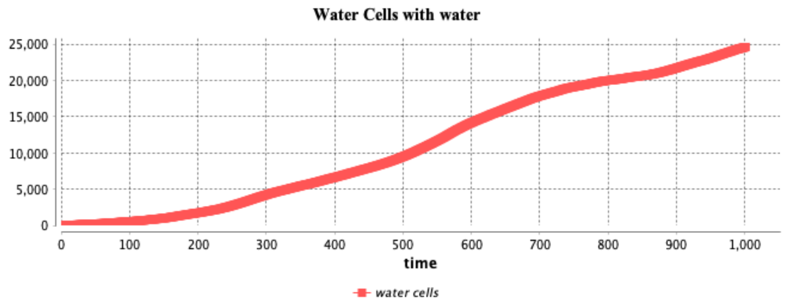


Figure 5: Cells that contained water

3 Assignment 3: Evacuation Model

3.1 Code Changes

From the `Goto` model, I incorporated the `continuous field of vision` model by adding the code chunk below and modified the heading so that the agents can rotate the vision up to 90° as per the instruction.

```
reflex update_perception {
    //the agent perceived a cone (with an amplitude of 60°)
    //at a distance of  perception_distance (the intersection with
    //the world shape is just to limit the perception to the world)
            perceived_area <- (cone(heading-45,heading+45)
            intersection world.shape) intersection circle(perception_distance);

  // if the perceived area is not nil, we use the masked_by operator
  // to compute the visible area from the perceived area according to the obstacles
    if (perceived_area != nil) {
        perceived_area <- perceived_area masked_by (wall,precision);
        }}
```

Another change was to measure the mean distance of the agents. Although the instruction required to use the `collect` argument, after encountering a few trial and errors, I ended up assigning a global variable named `mean_distance_to_goal` so that I can illustrate the mean distance on the chart.

Finally, I added a `stop_simulation` code that stops the simulation when all the agents leaves the room.

3.2 Interface

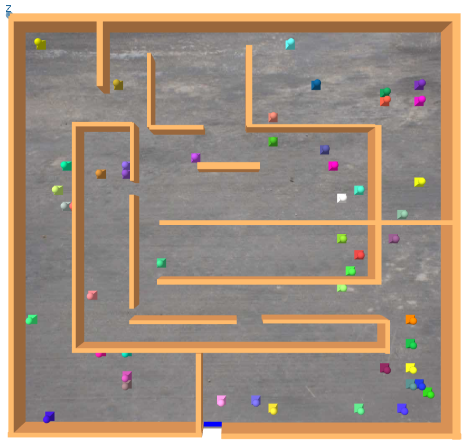


Figure 6: Map of the room

3.3 Results

Figure 11 illustrates the average distance between the agents' locations and the exit (target). The average distance among fifty agents began around 70 and increased to 90. Because the agents who started from the bottom left, where the Euclidean distance was closer to the exit, had to find an escape route that was further away from the exit, the mean distance increased. The mean distance quickly fell to 30 after it peaked. This is the first trough, as 12 agents have just exited or were preparing to exit the room. With a smaller group, the mean distance rose to 55 and then fell. The oscillation becomes smaller as the agents left the room.

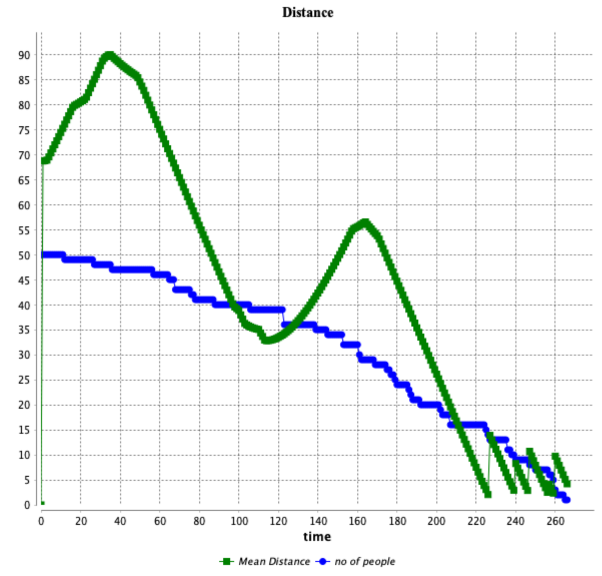


Figure 7: The mean distance to goal over time and the number of agents who haven't escaped the room

4 Assignment 4: Sensivity Analysis

4.1 The Task

The primary goal of this "Ants" simulation is to comprehend the parameterisation as well as the iterative process utilising '.gaml' codes. The idea in this model is that once the ants has food they set to diffuse the pheromon to either create chemical trails from their nest to promising food sources or releasing 'danger' alerts to nearby ants. Here, the evaporation parameter is used to change the time of evaporating the scent of pheromon.

According to the instruction, this simulation only tests the evaporation parameter, which ranges from 0.5 to 5 by 0.5 increment.

```
Evaporation: 0.5, 1, 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5
```

4.2 How to parameterise in GAMA

In GAMA, simply add an experiment to either visualise the simulation or run it in headless mode. It was only a few lines of code that needed to be changed.

```
experiment Repeated type: batch repeat: 2
keep_seed: true until: food_remaining <= 0.0 {
        parameter 'Evaporation' var: evaporation_per_cycle
        min: 1.0 max: 5.0 step: 1.0;
}
```

To export the file, we go all the way up to the `global` section and add the `save_result` section as follows:

```
reflex save_result {
          save ("simu:" +  int(self) + ";evaporation_per_cycle:" +
          evaporation_per_cycle + ";cycle:"+ cycle +
          ";food_remaining:" + food_remaining)
          to: "../results/results.txt" type: "text" rewrite: false;
    }
```

4.3 Results

4.3.1 Clean data

Here is a glimpse of the data frame. It contains of four variables `iteration`, `evaporation`, `cycle`, and `food_remaining`. The values of iteration ranges between 0-9, evaporation between 0.5-5.0 by 0.5 increment, cycle between 0-400, and food_remaining between 0-21.

```
library(tidyverse)
result <- read_delim("Assignment4/results/results.txt",
                     delim = ";", col_names = FALSE)

cleaned <-
  result %>%
  mutate(
    iteration = result$X1 %>% str_sub(6) %>% as.numeric(),
    evaporation = result$X2 %>% str_sub(23) %>% as.numeric(),
    cycle = result$X3 %>% str_sub(7,10) %>% as.numeric(),
    food_remaining = result$X4 %>% str_sub(16,20) %>% as.numeric()
  ) %>%
  select(iteration, evaporation, cycle, food_remaining) %>%  # clean variables
  arrange(iteration) %>%
  ungroup

cleaned
```

```
# A tibble: 30,912 x 4
   iteration evaporation cycle food_remaining
       <dbl>       <dbl> <dbl>          <dbl>
 1         0         0.5     0             21
 2         0         0.5     1             21
 3         0         0.5     2             21
 4         0         0.5     3             21
 5         0         0.5     4             21
 6         0         0.5     5             21
 7         0         0.5     6             21
 8         0         0.5     7             21
 9         0         0.5     8             21
10         0         0.5     9             21
# ... with 30,902 more rows
```

4.3.2 Food remaining by cycle

The first task is to understand how the evaporation affects the the time for food depletion. The 10 iterations were averaged for this outcome. Figure 8 demonstrates that the smallest parameter 0.5 extends food storage by the 1,356th time step. Parameters 1 and 1.5 have a similar effect keeping the food until the 740th time step, while 2 and above appear to be clustered between 250-400th time step.

```
## averaging the model iteration
cleaned %>%
  group_by(evaporation, cycle) %>%
  summarise(food_remaining = mean(food_remaining)) -> cleaned1

cleaned1 %>%
  mutate(evaporation = as.factor(evaporation)) %>%
  ggplot(aes(x = cycle, y = food_remaining)) +
  geom_line(aes(group = evaporation, colour = evaporation))
```

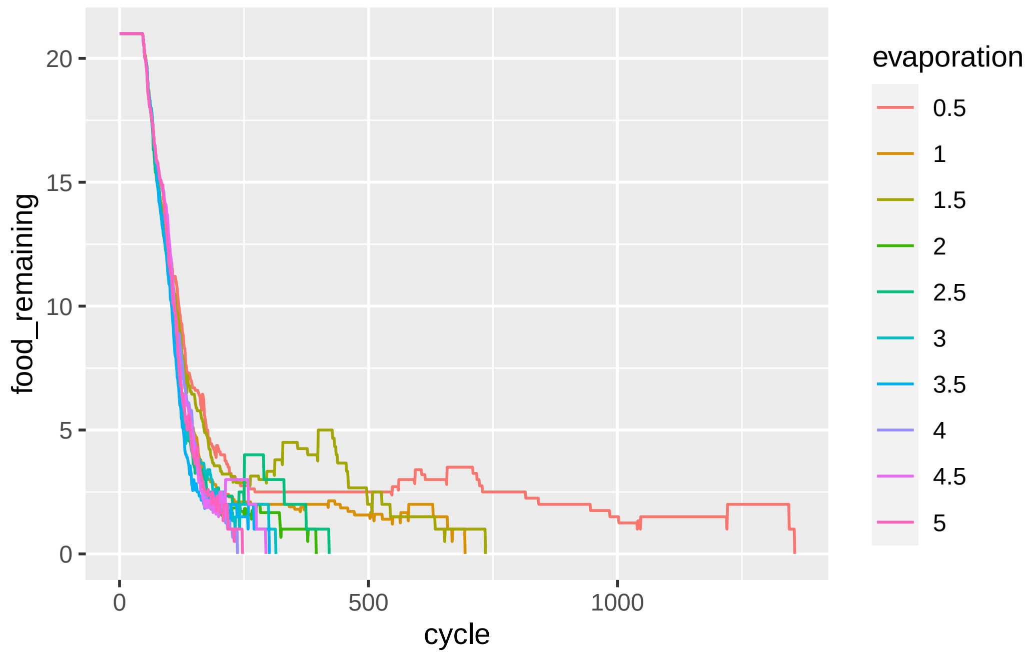


Figure 8: Time to food depletion by the evaporation parameter

4.3.3 Testing food depletion at every iteration.

Another interesting point is whether each iteration provides the same result. Indeed it does not show a consistent pattern. In Figure 9, it is shown that 0.5 significantly outweighed the time until food depletion in more than half of the executions. However, parameters 3 and above did not make a significant difference. As a result, it can be concluded that lower evaporation parameters slowed the time of food depletion in general, but this varied by iteration, and that higher parameter values did not make a significant difference in reaching food depletion.

```
cleaned %>%
  select(-food_remaining) %>%
  group_by(iteration, evaporation) %>%
  summarise_all(list(max)) %>%
  ungroup %>%
  mutate(iteration = rep(1:10, 10),
         evaporation = as.factor(evaporation)) -> sim_end

sim_end %>%
  ggplot(aes(x = evaporation, y = cycle, fill = evaporation)) +
  geom_bar(stat = "identity") +
  facet_wrap(~iteration, scales = "free_y") +
  labs(title = "Ants Model: Time to reach food depletion",
       subtitle = "by evaporation parameters and iterations") +
  theme(legend.position = "none",
        axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
```

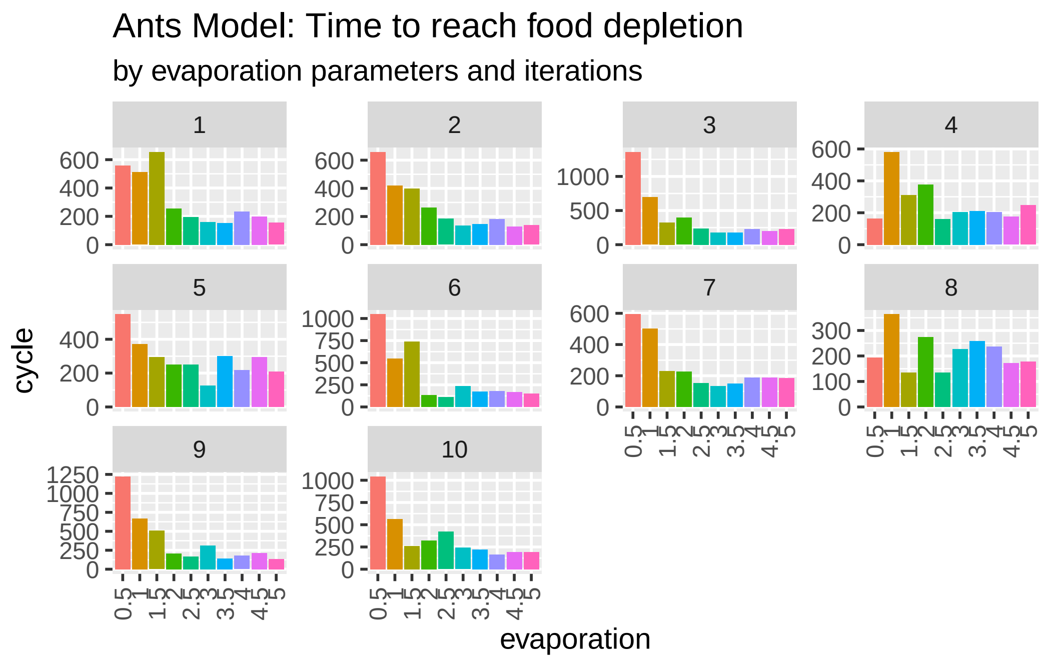


Figure 9: Time to reach food depletion by evaporation at each simulation run

5 Assignment 5: Dynamic species distribution model

5.1 How I extended the Eden growth model

- Creating `is_treeA` and `is_treeB` to germinate the seeds
- Switch the seedling function
- Adding competition between tree A and B

5.2 A Summarised ODD Protocol

The purpose of this agent-based model is to examine the spatial and temporal patterns of tree growth and to make a preliminary estimate of tree competitions. Specifically, we are addressing the following question: how does trees grow and distribute in the environment in response to tree (nature) competitions.

The model includes the following entities: trees and land. The state variables and attributes characterising these entities are listed in Table 1. The environment is built on a grid of 100 x 100 tessellated patches. Unlike in the real world, seedlings can grow and spread with the same probability in this virtual space. Because it is a closed environment (rather than a taurus), the trees do not spread beyond the boundaries. There are two types of trees in this model (seedlings). I categorised them treeA and treeB for convenience of reference. Both trees are placed in a grid cell at random, but not in the same cell. Since they are seedlings, they are not allowed to move, which means that once they are rooted to the ground, their growth are dependent on the environment. As for the temporal and spatial resolution, a time step in the model represents a year and the simulation stops at the 400th cycle. There is no particular representation for the spatial resolution.

Table 1: State Variable

Name	Variables
is_tree	true / false
is_treeA	true / false
is_treeB	true / false
is_seedling	true / false
treeAge	integer (starts with zero)
maxAge	Somewhere between 80-120

The model's most important processes, which are repeated at each time step, are the updating of germination, growth, tree distribution, and competition with other tree types. Each tree germinates with a 75% chance within 5 years (i.e. cycles in the GAMA platform), and if successful, spreads using the von Neumann neighbourhood method. If tree types A and B overlap, there is a competition between them, but A will win by 70%.

The model's most important design concepts are seedling competition and ageing. In terms of seedling competition, all cells that do not contain any trees have an equal probability of germination. However, if a tree has already grown in a cell and comes into contact with another tree seedling, we have a competition. Tree A has a 70% chance of beating tree B. When it comes to seedling ageing, if a tree reaches its maximum age, which is a random number between 80 and 120, it dies.

5.3 Results

Figure 10 shows the result after the 400th cycle of the simulation. The two seedling types are noticeable, as is the spread, and there are some signs of competition in between. Figure 11 depicts the temporal increase of trees graphically. Up until the 80th step, the increasing trend was very steep. This is when the competition and new germination comes into place. The temporal oscillations of the tree types appear to be similar. At the end of the simulation the number of trees A is 6,396 and the number of trees B is 4,945, where A is approximately 29% higher than B.

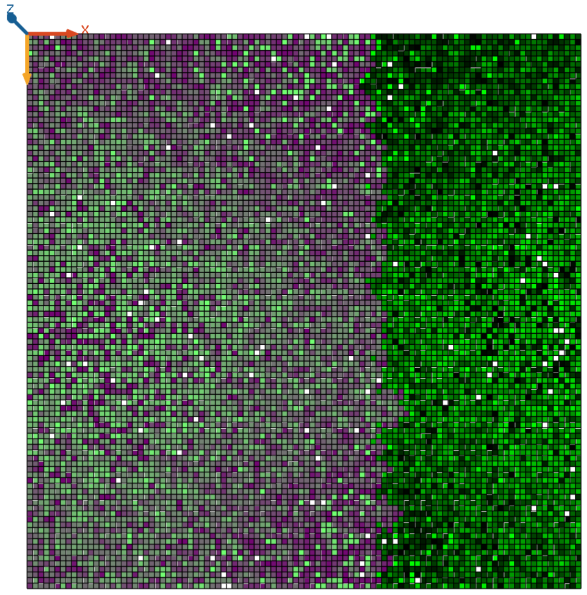


Figure 10: Forest Status after Cycle 400. It shows that there a Competition between Tree A (green) and Tree B (Purple)

Since it was assumed that tree As would win the competition by 30%, tree As won the majority of the repetitions. However, depending on where the initial seedling was planted, the opposite was also observed. As a result, while tree As are statistically proven to have higher tolerance than their competitors, this may not always be the case.

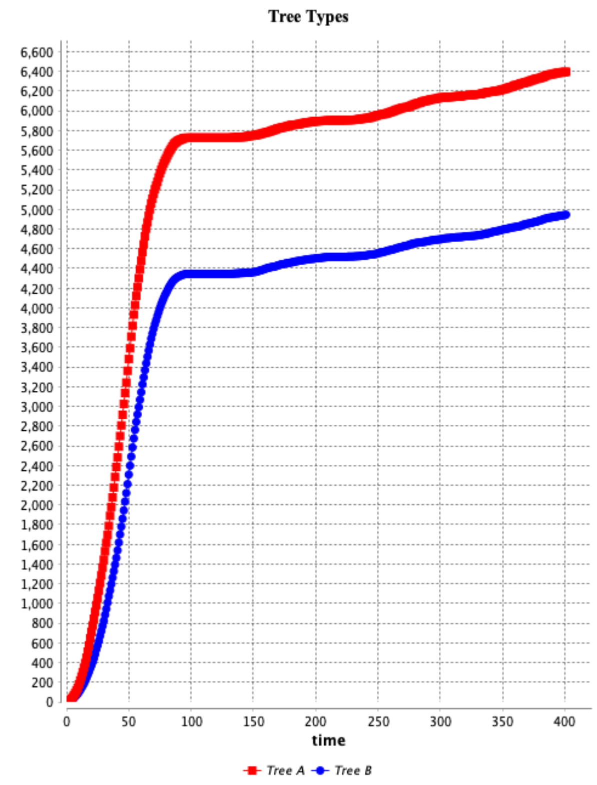


Figure 11: A Graphical Representation of Tree types A and B over time

Acknowledgement

I thank Dr. Gudrun Wallentin and Christian Neuwirth for their kind support on my enquiries.

Data and Code

All of the data and code are stored in my GitHub repository https://github.com/dataandcrowd/UNIGIS-Models. You can use this to explore the data or run the simulation on your own computer.