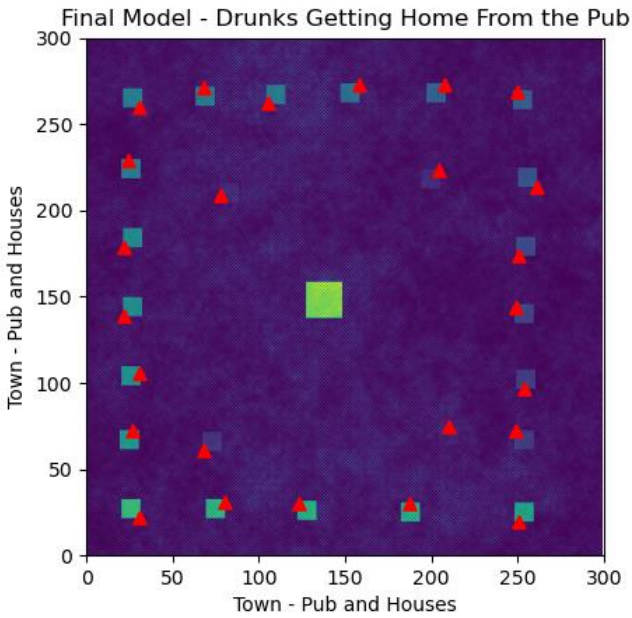
Programming for Social Scientists: Core Skills Assignment 2

Planning for Drunks – Supporting Document

The purpose of the document is to give a brief context of the software, explaining how it ended up how it is so that any issues can be understood and marked appropriately. The software is developed in Python 3 using the IDE Spyder.

1. Intention of the Software

The intention of the programme is to simulate a random walk of 25 drunks which stops when the drunks reach their allocated home. The software pulls in a raster file that identifies the pub point and the houses, and then creates 25 drunks who start at the pub and then make their way home. The aim of the software is that each drunk will reach their home (not necessarily in the most efficient way) and the software will then plot the drunks at their respective homes. The software lets the user know when each drunk reaches their house, and the time it took for the drunks to get home. Density mapping is used to show which paths the drunks took, and at which points the drunks passed over the map the most. The user is then able to save the new density map as a txt file.



2. Issues During Development

As is, the code runs with no known issues. A couple of issues arose during the development process, most but not all of which are solved.

The first issue was that the raster file did not initially display the pub when pulled in. To overcome this a condition was added when reading in the environment to change the pub number from 1 to 300 to ensure there was more contrast on the plot and it stood out. This problem persists for some of the houses closer to the pub, where it is hard to distinguish them from the background.

A second problem during the development was that when density tracking was included in the model, this removed the houses from the plot. At first there only seemed to be the option of a density plot with drunks but no houses, or a plot with houses and drunks but no density mapping. To overcome this the raster file for environment was read in a second time and a separate list created for density which was used in the 'mark' function to mark density on the plot. The outcome was that it worked but the density is a lot less distinctive than it would have otherwise been.

Many iterations are required to get the drunks to their homes which limits the efficiency of the code. The current number of iterations is set at 100,000 but the number is reduced to 10,000 (for example) the drunks do not get home. In an ideal situation I would prefer the model to run in a way where the drunks get home in the most efficient number of iterations without that number having to be set. However, given that they are 'drunks' and it is a random walk this is not necessary for the model.

Ideally the model would have been animated to track how the drunks move and show the user how the drunks get from the pub to their houses. When animation was attempted the model would be plotted, but no animation would occur, and the IDE would often crash. It was decided to omit the attempt to animate to prioritise smooth running of the code.

3. Sources

A couple of different sources were used in the development of the code.

Some parts of the code are directly referenced from my own code which was used in the previous assignment of building an ABM. In particular, the code for reading in the csv file is and the creation of the walk home function are like what was created in the ABM model. Additionally, the code for marking the density is like the code that marked what areas the sheep ate in the ABM practical.

I have also referred to stack overflow on a number of issues, in particular for setting up the timer and the code for the perf_counter().

I also reviewed some YouTube videos on reading in csvs, random walks and on creating UMLs.

I have been referring to Eric Matthes' (2019) book 'Python Crash Course' throughout the development process. I looked up random walks, proper documentation of functions, print statements, and testing as well as practising different elements of code by using the examples in the book.

4. Development Process Including Tests

The development process had four main parts, focusing on each of the key requirements of the project.

1. Setting up the town

The first part entailed pulling in the data and finding out the pub points and home points and drawing these on the screen. To do this the csv module was imported to allow the reading of csv files. A list for the town was created and each row of the raster csv file was appended to the town environment. Within the raster file there was numbers which marked the positionality of the houses and of the pub. Matplotlib was then imported to display the environment and as mentioned earlier the pub did not show up. The if statement to recognise the value 1 (which was the pub) and return this as 300 was added which allowed the pub to become visible when plotted. The first part was completed as the houses and the pub were all successfully drawn on the screen.

2. Making the drunks and allocating them houses

To create the drunks a drunk framework was created which included a drunk class. Within the initial drunk class, there was drunks and their starting position of the pub in the plot (x=137, y=158). The drunk framework was imported into the main model. A list was created to add the drunks to the main model and the drunks were appended to the list. A test plotting was then run to make sure that the drunks were at the pub to start with.

After this the agents were allocated houses. The drunk class was amended to include attributes for their houses and were called in a for loop in the main model which allocated each drunk a house based on the positions of the houses of the graph (10, 20, 30 etc.). These positions were then appended to the drunks list. To check that the houses were correct a print statement was ran after the allocation of the houses to make sure each drunk had been allocated a house as expected.

At this point I was satisfied that the drunks had been created, were starting at the pub, and had been allocated a house number.

3. Moving the drunks and drawing their moves to track density of steps

To move the drunks an additional function was added to the drunk framework. The walkhome function was added to simulate a random walk, which uses a random number generator and, depending on whether that number is lesser or greater than 0.5, the drunk will move one step in a direction. This function was called in a for loop in the main model which determined that if the co-ordinates the drunk reached within the town was not their house, continue the random walk. However, once the co-ordinates of the drunk matched their house, then stop the walk and print a statement that told the user that the drunk was home. This was tested by running the print statement and plotting the drunks and the town to see if they had moved to their houses.

Once satisfied that the drunks were made and were able to get home from the pub, it was time to mark the density of their walks. To do this an additional function was added to the drunk framework which added 1 to each step taken by the drunk to track where they had walked on the town map. This function was added into the for loop which moved the drunks to ensure that when they moved their movements were marked. At this point the problem arose whereby the plot was only showing the drunks at their home co-ordinates and the density mapping, but not the homes and the pub as highlighted in the development problems section. As mentioned, a second reading of the csv was completed to create a list for the density plotting, and the mark function called on this new csv reading density rather than the original town mapping. This solved the problem.

At this point there was a clear mapping of all the drunks in their houses and the density of the steps was mapped on the plot and there was an unsuccessful attempt to animate the model. A timer was added to the programme so that the time taken for the drunks to get home could be monitored, with the usual time sitting between 11-15 seconds.

4. Saving the density file as a txt file

To save the new density plot as a txt file the density data was opened using the csv module, and the new data was written into the file row by row and saved as a txt file. The txt file can be retrieved from the ‘Files’ element of the console under ‘density.txt’. In the file the agent’s movements within the raster csv data can be seen.