Documentation: Planning for drunks - Show me the way to go home...

1. Summary

Version 1.0.0

This documentation aims to provide an overview of the agent-based model, DrunkModel.py, that was developed for Assessment 2 of GEOG5990M Programming for Geographical Information Analysis: Core Skills.

The model simulates a town of 25 drunk people trying to "stumble" back to their homes from the pub. It outputs a density map and .txt file to show where drunks stumbled on their way back to their homes.

It was developed in Spyder 3.3.2 (Python 3.7) and the following documentation describes how to run it from this IDE.

There are no further plans for development of this program.

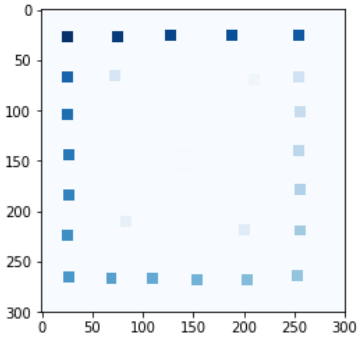
2. How to Run

Files required to run the model:

- DrunkModel.py – this is the main model from which the program is run,
- Drunkframework.py – this contains the Drunk class and outlines the characteristics of the drunks,
- town_plan.txt – This is a 300 by 300 raster file representing the town in which the drunks stumble about in. The pub is denoted by 1s, the houses by the numbers 10-250, in increments of 10, and everywhere else, zeros.

These files can be found on my GitHub:
https://github.com/hannahwh05/Assessment2

In Spyder set Tools > Preferences > Ipython console > Graphics > Set backend to inline.



For this model, town_plan.txt has been used for the environment (Figure 1). The code allows for an alternative csv environment to be imported.

Figure 1: Town Plan

The simulation is run from tkinter GUI. When the code is run, a window will appear on the computer screen called Drunk Model. To run the model, click "Run" from the "Menu" in this window, as shown in Figure 2.

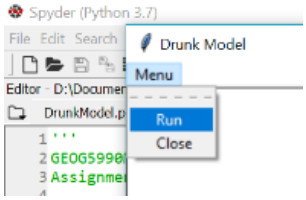


Figure 2: How to run the model

When the model has met the "stopping condition", where all the drunks have arrived home, as shown in Figure 3a, close the window and a density map showing the points where the drunks have passed through, will be printed to the console and saved as density.png to the current directory, shown in Figure 3b, along with density.txt file containing the density points. 1 is added to every point the drunks pass through.

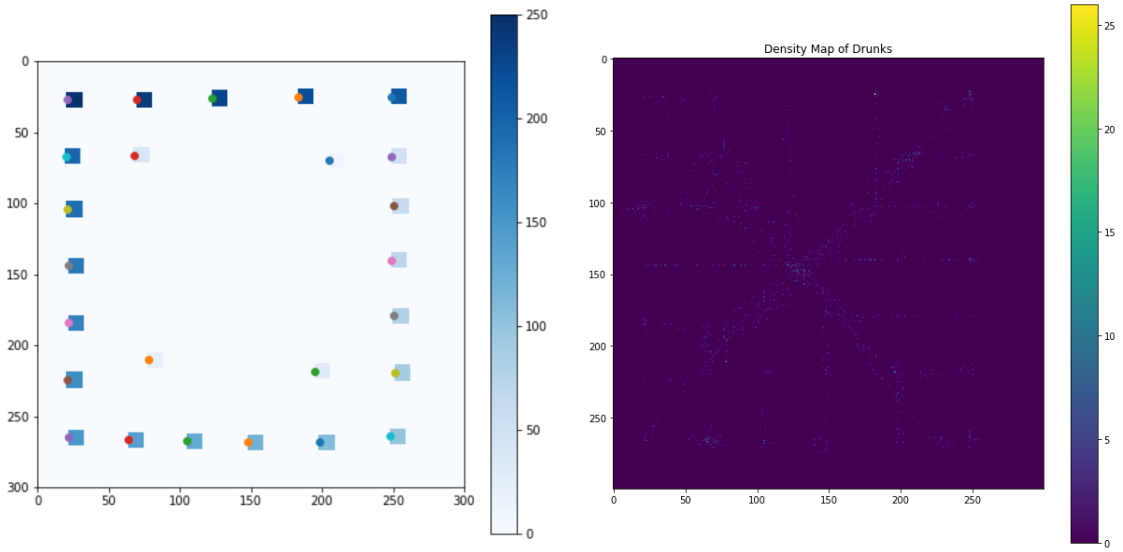


Figure 3: a) End of Simulation (left) and b) Density Map of Drunks (right).

3. Program Description

This program does the following:

1. Pulls in a csv/txt file with values for an environment and finds out the pub point and the home points.
2. Displays the pub and homes on the screen.
3. Creates drunks and assigns them a home each.
4. Animates the drunks leaving the pub and reaching their homes.
5. Stores how many times drunks pass through each point on the map to a density.txt file.
6. Displays the density of drunks that have passed through each point on the map and saves it to density.png.

There are 25 drunks, each of which have homes assigned to them according to their number (creation number plus 10 (to match the house number). Houses are labelled 10-250 in increments of 10. The drunks randomly “stumble” left, right, up and down, back to their homes from the pub based on conditions met in a for loop. An element of bias is introduced to give the drunks some direction to their homes and prevent them from retracing their steps (though there is a small probability that they can to increase the randomness of their movements, as described below). For every point that the drunks stumble through on the map, 1 is added to the corresponding value in the density.txt file and therefore the density.png. When the drunk reaches their home, they stop stumbling. They print to the console their house number, door coordinates and announce, "Finally I'm back! Off to bed!". Once all the drunks have arrived back home, "Stopping condition has been met. All drunks are home!", is printed. At this point the animation window can be closed triggering the density data and map to be saved to the current directory.

Certain elements within the model and framework can be changed to suit the user:

- The environment/town_plan – this is imported based on the numbers of rows and columns in the file. Provided the buildings are square, the coordsFinder function will locate the pub, houses and their doors based on the ID inputs. n.b. if the house numbers are not in the 100s, alter line 129 in DrunkModel.py as appropriate.
- The number of drunks – consequently to the above point, the number of drunks can be changed to match the number of houses within the imported environment.
- The degree of drunkness/randomness of the stumbles – this is dictated by:

    - The "randomness" variable on line 156 of the DrunkModel.py where, randomness = random.random()/3, giving a random number of uniform probability between 0 and 0.3 recurring. The higher the random number the more drunk the Drunk is, where 0 would lead the Drunk directly to their home.
    - the "unitsMoveBy" variable, on line 88 of the drunkframework.py, where unitsMoveBy = random.randint(1,5), giving a random integer between 1 and 5.
- The names of the txt and png files can be changed in lines 283 and 301 respectively.

4. Development and Issues

This program was developed using code from my Assessment 1 agent-based model (ABM) for this module. That model can be seen on my GitHub repository at: https://github.com/hannahwh05/GEOG5990M_Programming.

During development of the model, I aimed to make it as adaptable as possible so that a different environment could easily be imported and consequently the number of drunks could be changed. Frequently, this meant starting with a specific function suited to this project and then generalising from there.

Initially, I wanted to find the coordinates of the pub exit so I could determine the drunks' starting point. However, I then realised this could be adapted to find any set of coordinates for doors within the environment, provided a value or "ID" was input. I could therefore use this function to find each drunk's house door.

When defining the way my drunk moved, I wanted to make it clear that their movements were random and that they were not just going directly to their homes, so I called the function "stumble". To start with I made the function very similar to my "move" function from my ABM, but with the addition of a condition where it could only move if it was within a value of the environment that was equal to zero. I soon realised then that they would not be able to arrive home as their house values would not be equal to zero. I also adjusted the torus to create the appearance of town walls so that drunks could not just cross space and time!

This version was working for a while during the simulation but then it would break, due to "list index out of range". Ultimately the drunks would take a very long time to reach their homes as they wondered aimlessly around the environment. I thought of a way of giving them direction by creating a function that would calculate the distance to their homes. At first, I just input the pub door coordinates but thought that the drunks would need constant updates on their distance to make

it home. As a result, I changed this to general X and Y coordinates that could be input. I then defined the current and potential distances within the stumble function. This made the drunks movements less randomised, so I introduced a "randomness" variable to the stumble function, making them appear more drunk.

When updating Spyder to 3.3.2 I found that despite the backend being set to inline, an additional blank window, called Figure 1, was being created every time I ran the model. When matplotlib.pyplot is mentioned in the code it creates a popup window. To resolve this I have added the following lines to a couple of point in the code so that plots are printed to the console:

```
160 #set map inline
161 shell = IPython.get_ipython()
162 shell.enable_matplotlib(gui='inline')
```

Figure 4: Code to set figures inline

Due to time constraints and difficulties adjusting specific colours in the environment, I was unable to overlay the density map on top of the animation.

5. Potential Additions to the Model

To improve the model, a custom colormap could be created to form discrete colours for each value within the environment, as currently the pub cannot be seen within the simulation as it is equal to 0. The colours of the drunks could also match the colours of their houses. Ideally the density map would be overlaid on top of the animation if 0 in the environment was transparent.

Additionally, distractions could be added along the way, e.g. a chip van, so they wander off course and don't head straight home. Pathways could be added so the drunks have to walk in certain ways.

Finally, the code could be further optimised to improve run time, for instance, by only calculating distances for moves that are actually made rather than for all potential moves. However, the length of run time has not been an issue at present.

6. Information about the Developer

I am a current University of Leeds MSc GIS Student (2018-19) learning core skills in Python. I graduated from University of Southampton in 2016 with a 2:1 in BSc Geography.

Link to my website: https://hannahwh05.github.io/