Warwick iGEM computational modelling

Table of contents

Contents

Custom modelling

The most significant component of the modelling we performed as part of the project was the design, implementation, and interpretation of a custom stochastic model we designed to show that our proposed product would be "useful" in the real world.

We understand that computer models can become quite dry, particularly when explaining the details of their implementation, as if this is not done precisely, a small misunderstanding can be quickly amplified to an unexpected result, as the model's high complexity causes it to have a fairly so-called chaotic output.

As a result of this, we created an in-browser interactive implementation of the model, which plots the output graph of the model based on the user inputting initial parameter states. We intend that this can quickly, intuitively, and interactively show the function and results of the model, which can help inform the goal throughout the implementation explanation, and provide a top-level understanding even if the rest of this page were omitted.

Introduction

To show that our proposed product would positively benefit the environment where it is proposed to be used, we wrote a computer model of the environment, with and without the product in use, and showed that when it is in use, the model scenario result improved.

Abstract We propose a validated computational model of the spread of an antibiotic resistant pathogens in a hospital, with and without our diagnostic tool for quickly identifying it, and show that in a relevant scenario it reduces the presence of antibiotic resistant pathogens in our selected scenario, showing our product is beneficial in the real-world.

Motivation The purpose of the model is two-fold:

- Demonstrating that our product is beneficial
- Understanding the use cases where it is most and least applicable

Assets The whole project repository is available on GitHub

The final production code for the project can be found here:

- As a standalone python file

- As a package on PyPI

Model development

Model type Our model is discrete time, stochastic, and compartmental:

- Compartmental means that the model is expressed in terms of the transitions between a set of states. The logic for these transitions forms a fundamental part of the model
- Stochastic means that the model is based on random probabilities, as opposed to a deterministic system of equations
  - A set of constant probabilities define the properties of the model
  - Transitions between states are chosen randomly with these constant probabilities

  These probabilities, and other variable aspects of the model, such as population size or how many drugs are used, are set as constant values at the top of the model.

  Initially, the model just had a parameter for how many different antibiotics are used, and all the associated probabilities (e.g. likelihood of recovery, likelihood of death, etc.) with these antibiotics were the same, but in the final version, the different antibiotics are named to more closely map to the real world, and they are allowed to have their own separate values for these probabilities. However, for convenience's sake, we introduce meta-parameters which can be used to set all the antibiotics to have the same probability in a given category.
  Below shows code for a default setting of these probabilities, the meaning of which will be explained further on:

```
# General model parameters
NUM_TIMESTEPS = 100
POPULATION_SIZE = 500
INITIALLY_INFECTED = 10

# Ordered list of drugs used, their properties, and the properties of their
# resistant pathogens
DRUG_NAMES = ["Penicillin", "Carbapenemase", "Colistin"]

PROBABILITY_MOVE_UP_TREATMENT = 0.2
TIMESTEPS_MOVE_UP_LAG_TIME = 5
ISOLATION_THRESHOLD = DRUG_NAMES.index("Colistin")

PRODUCT_IN_USE = True
PROBABILIY_PRODUCT_DETECT = 1
PRODUCT_DETECTION_LEVEL = DRUG_NAMES.index("Carbapenemase")
```

```
##############################################################
# Use these if you want to set all drugs to the same thing #
##############################################################

PROBABILITY_GENERAL_RECOVERY = 0
PROBABILITY_TREATMENT_RECOVERY = 0.3
PROBABILITY_MUTATION = 0.25
PROBABILITY_DEATH = 0.015
# Add time infected into consideration for death chance
DEATH_FUNCTION = lambda p, t: round(min(0.001*t + p, 1), 4)
# TODO: Make this more robust
PROBABILITY_SPREAD = 0.25
NUM_SPREAD_TO = 1

##############################################################################
# Set these explicitly for more granular control, or use the above to set #
# them all as a group                                                        #
##############################################################################

# Lookup table of drug properties by their names
DRUG_PROPERTIES = {}
DRUG_PROPERTIES["Penicillin"] = (
    PROBABILITY_TREATMENT_RECOVERY,
)
DRUG_PROPERTIES["Carbapenemase"] = (PROBABILITY_TREATMENT_RECOVERY,)
DRUG_PROPERTIES["Colistin"] = (PROBABILITY_TREATMENT_RECOVERY,)

# Lookup table of resistance properties by their names
NUM_RESISTANCES = len(DRUG_NAMES)
RESISTANCE_PROPERTIES = {}
RESISTANCE_PROPERTIES["None"] = (PROBABILITY_GENERAL_RECOVERY, PROBABILITY_MUTATION, PR
RESISTANCE_PROPERTIES["Penicillin"] = (PROBABILITY_GENERAL_RECOVERY, PROBABILITY_MUTATI
RESISTANCE_PROPERTIES["Carbapenemase"] = (PROBABILITY_GENERAL_RECOVERY, PROBABILITY_MUT
RESISTANCE_PROPERTIES["Colistin"] = (PROBABILITY_GENERAL_RECOVERY, PROBABILITY_MUTATION
```

Additionally, there are internal settings, for example how the model outputs it results.

- Discrete time means that changes in the model occur at granular timesteps - like turns in a boards game

  Below shows the code for how operations are performed on every person in the population each timestep, and data about them recorded

```
# Make a new data handler for each simulation
self.data_handler.__init__()
```