The Artificial University v 1.1: Open Source Release

Project title

The Artificial University (TAU) – A Decision Support Tools to Get back to Campus and Cope with Covid for College and University Administrations.

Motivation

The Artificial University (TAU) is the strategic policy experimentation and visualization tool within The MITRE Corporation's University Toolkit for operating universities in the COVID-19 era (MUT). MUT offers several tools in addition to TAU: a decision dashboard, early-warning sewage analysis, COVID-testing, smart tracing, and vaccine management. TAU's role in the toolkit is to give university administrators a highly configurable artificial university on which to run policy experiments. TAU facilitates the evaluation of intervention plans using multiple metrics, generates insights to inform policy decisions, and produces visualizations that help to communicate policy reasoning. TAU is a uniquely useful epidemiological model because: (1) it takes account of human factors such as social networks, individual vulnerabilities, and multiple types of compliance; (2) it is highly configurable to individual universities and accommodates politically feasible interventions and varied health and equity metrics; (3) it is calibrated using only public data, while allowing individual universities to employ confidential information if they so choose; (4) it can be made responsive to regularly updated data about the local region and to university health data; (5) it supports cost-benefit analyses of a host of infection and anti-body testing strategies; and (6) it is open-source and benefits from open-source collaboration efforts.

Build status

TAU builds and runs in version 1.1, the subject of the open-source release on July 15, 2020.

Code style

This version of TAU was not developed with strict guidelines but is generally intended to make the code as readable as possible.

The in-line documentation style uses informative method names and local variable names, and in-line comments for method parameter names (i.e. methodCall(/*parameterName=*/

parameter)). The aim is to make the code as clear as possible without comments, only requiring comments when the code purpose cannot be made clear with clear code and naming alone.

Indentation is used as much as needed to make the code as readable as possible.

Braces are used after if statements even when not necessary for code compilation or function.

Agent and class names use PascalCase. Method and variable names generally use camelCase. However, casing was inconsistent in model development, and some older methods and variable names use PascalCase, snake_case, or other case methods. Generally, we stick to the usual Java convention of PascalCase for class names and camelCase for non-constant method and variable names, with some exceptions being made for parameters that may be visible to non-coder users of the model.

Framework used

AnyLogic version 8.5.2 (https://www.anylogic.com/) is the platform used for the model and simulation development, so some AnyLogic frameworks are used. For the agent-level SIRS model, statecharts are used to represent the person's status regarding infection, isolation, and testing. Inheritance is used for Student, Faculty, and Staff agent types to inherit the behavior of the Person agent base class.

AnyLogic's message-passing framework is used for people agents to infect each other, administer tests, and contact trace.

AnyLogic events are used for a daily infection and testing job.

Java I/O from the java.io.* package is used for file reading and writing.

Features

Details about the model implementation and parameters are in the ODD-D documentation, released with TAU.

Code Example

Below is a coding example of the code which assigns on-campus housing buildings to dining halls. Note that this version of the model (TAU v 1.1) assumes that buildings are assigned to dining halls, and that all students who live in a building will only go to its assigned dining hall.

We first calculate an approximate number of people who will be assigned (indirectly, through their building) to a dining hall - aiming to make the number of students assigned to a dining hall approximately even. Then, we randomly assign buildings to a dining hall until the approximate number of people is reached, then move on to the next dining hall. Note that the random

initialization uses the Random object initializationRandom, which is set for all initializations in order to keep a consistent network across simulation runs.

This code is found in the method Main#makeDiningHalls.

```
int buildingIndex = 0;
int approxNumPeoplePerDiningHall = (int)(studentPopulation.stream()
.filter(p -> p.livesOnCampus).count()
/ ((double) numDiningHalls)) + 1;

int diningHallNum = 0;
Collections.shuffle(housingBuildings, initializationRandom);
while (buildingIndex < housingBuildings.size()) {
	DiningHall hall = new DiningHall(diningHallInfectivity);
	hall.setName("Dining Hall " + (diningHallNum++));
	allConnections.add(hall);
	 int numPeopleAssignedToCurrentDiningHall = 0;
	 while (numPeopleAssignedToCurrentDiningHall <
approxNumPeoplePerDiningHall
			&& buildingIndex < housingBuildings.size())
	{
		hall.addBuilding(housingBuildings.get(buildingIndex));
		numPeopleAssignedToCurrentDiningHall +=
housingBuildings.get(buildingIndex).getPeople().size();
		buildingIndex++;
	}
}
```

Installation

AnyLogic must be installed to run the model. Most functionality of the model can be run in the free Personal Learning edition of AnyLogic version 8.5.2, which can be downloaded at https://www.anylogic.com/downloads/ and can be installed on Windows, MacOS, and Linux.

API Reference

AnyLogic documentation can be found at https://help.anylogic.com/index.jsp.

Tests

This version of the model includes an AnyLogic Experiment called SimulationSchool, which allows you to adjust the parameters of the model and perform a single run and view the infection curve over time. We have performed face validation of the core of the model (without the TAU network), and have rechecked that face validation on extreme scenarios in SimulationSchool.

How to use?

To run a single simulation and view the infection curve, the AnyLogic experiment SimulationSchool allows the user to adjust many parameters and perform a single run of the simulation. This can be done by opening the model in AnyLogic and navigating to TheArtificialUniversity > SimulationSchool. After clicking SimulationSchool, a properties pane will open, and model parameters may be adjusted. While many parameters are adjustable in the SimulationSchool parameters, some of the highly customizable university network parameters are hard-coded into the model. These parameters can be found in the initializeSchool method of the Main agent. We include two sets of parameters for the university network – the large research university and the small college. This setting is toggled by the top-level parameter IRU, IsResearchUniversity.

To run a parameter sweep, we can use the experiment SchoolExperiment. Directly cloning from the git repo, one should be able to immediately run SchoolExperiment on the parameter sweep included. The SchoolExperiment experiment includes the Java I/O to read the parameters from the included file, TAU_parameter_student_compliances_samples_count_510.csv, and output results to the files AggregateResults.csv and AggregateResultsDaily.csv. Note that simulation runs of the large university cannot be run on the personal learning edition of AnyLogic, as it uses too many agents. Also note that it may be necessary to increase the maximum available memory in order to run the simulation; this can be done in the properties pane of the SchoolExperiment experiment.

MVP Version

TAU version 1.1 (released as open-source code on July 15, 2020) is being developed into Minimum Viable Product version, which should be available at or near the end of July 2020. This freely available product includes a user-friendly interface that addresses a dataset produced by TAU, enabling users to navigate and visualize findings without needing to employ the AnyLogic platform or even run the model.

Contribute

This is an Open Source collaboration project. Any improvements of the current methods as well as adding new methods is encouraged. Any updates should be send with supporting literature and references to tauopensource@gmail.com. The Human Simulation Group, which is releasing TAU version 1.1, is also working on more advanced versions; suggestions and ideas are welcome.

Credits

The Core Version was developed by the Human Simulation Group under lead of the Center for Mind and Culture in Boston with active contributions of the Virginia Modeling Analysis and Simulation Center at Old Dominion University and conceptual recommendation by The MITRE Corporation under the umbrella of the Covid-19 Healthcare Coalition (https://c19hcc.org). The code was reviewed and improved by Simudyne.

License

Human Simulation Group
Center for Mind and Culture
http://mindandculture.org/