Catalyst: Agents of Change

Shannon Gallagher

January 19, 2018

1 Outline of software

```
INPUT: Control file with agents file; disease parameters;
 disease model; time steps; paralllel options;
 activity schedule; individual agents/instantiations to track
for every timestep t
  for every activity at time t
    prob_infection = infect_agents(agents, activity,
                     time, disease)
  end
  agent_status = update_agents(prob_infection)
  update_activities()
  update_disease()
end
OUTPUT: status of agent at each time step, saved as multiple text files;
 number of agents in compartment at each time step;
 specific agents over time; activity info
```

1.1 Infect Agents

```
INPUT: agents, activity, time, disease
for every infected agent ii
  for every neighbor of agent ii, jj
    update_infection_prob(ii,jj, activity, disease, time)
  end
end
OUTPUT: updated agent probabilities
```

2 Description

2.1 General Outline

The benefit of this program is that the agent-based part (the infect_agents portion) is completely modular. The function update_agents() is a multinomial

draw. Thus this portion can be used both for heterogeneous individual agents along with homogeneous agents (the compartment-model).

2.2 Infect Agents – the "AM" portion

This function is what differentiates the model from a CM. In this step, agents explicitly interact with one another and hence can change one another. This interaction has the potential to be very slow. Naively, if we were to loop over all pairs of agents and all environments then we would be looking at $O(N^2E)$ runtime where N is the number of agents and E is the number of environments.

Hence, it is critical to be speed up this step. It is worth noting that in SI-framework, the only interactions that are worthwhile is when an agent is actively infecting another. All other compartment transitions are independent from one another, meaning they do not depend on the status of other agents. Hence we only need to loop over agents in "infectious" states. Moreover, we only need to look at the neighbors' of these infectious agents, specifically those in susceptible states.

We also assume that once an agent is infected by another, she will be unaffected by further infections. As such, once an agent is infected by another, we remove her from the susceptible pool.

We plan to reduce this step to $O(I)$ where I is the number of infectious individuals where, typically $I << N$. The first step is that we only need to loop over infectious agents, for a run time of $O(I)$. We then need to loop over all neighbors of infectious agents. However, we note that we can precompute potential neighbors and store them in a dictionary, so look up will be $O(1)$. Neighbors will be over all environments so we no longer have to loop over those. Thus we only have to check of neighbors of the infectious are susceptible and update the infection probabilities based on the environment activities, two agents, and disease parameters.

3 Curent implementation

- A working framework in C
- Incorporation of GNU Standard Library packages for ODE integration and random number generation
- A working SIR model
- Testing framework to initialize agents and environments
- A Dictionary of neighbors through GLib's GHashTable

4 Next Steps

- infection to change probabilities of agents.

  - How to store these? CM probs vs. agent probs

- integrate different modules to simulate the SIR

5 Long Term

- Input file standards
- Make the ODE solver more generic to support general systems. The main problem is how to incorporate this symbolically to be an easy input for the control file
- Estimate ODE parameters from an AM through least squares or mle.
- Convert SPEW agents to a viable format for catalyst
- Make nice interface for control file (shiny)
- Visualize outputs

6 Schedule

6.1 DONE *<2018-01-01 Mon> - <2018-01-13 Sat>*

- Set up preliminaries.
- Learned how to use C libraries, especially GSL (GNU Scientific Library).
- Got a much clearer demonstration of how pointers work
- "Finalize" framework for software

6.2 DONE *<2018-01-15 Mon> - <2018-01-21 Sun>*

6.2.1 The "AM" part

- set up testing framework for cm-am
- Work and get used to GLib and included data structures including Arrays which can grow, singly-linked lists, and hashtables to use within C.
- Made a function to make a dictionary of neighbors.

6.3 TODO *<2018-01-22 Mon> - <2018-01-26 Fri>*

- Store both a single probability and multiple probabilities for agents, ... likely not stored at each step
- Start work on fitting a SIR model to the AM
- Visualize outputs in R