IONTW Reference Guide

Version 1.1

Winfried Just* Hannah Callender† Drew LaMar‡ Natalia Toporikova§

September 1, 2014

Contents

*Department of Mathematics, Ohio University, Athens, OH 45701 E-mail: just@math.ohiou.edu
†University of Portland E-mail: Callende@up.edu
‡The College of William and Mary E-mail: drew.lamar@gmail.com
§Washington and Lee University E-mail: toporikovan@wlu.edu

1 Infection Model Types

- SIS: **gain-immunity** = **Off** and **latent-period** = **Off**
- SIR: **gain-immunity** = **On** and **latent-period** = **Off**
- SEIS: **gain-immunity** = **Off** and **latent-period** = **On**
- SEIR: **gain-immunity** = **On** and **latent-period** = **On**

2 Disease Transmission Parameters

- `<tau>`: Mean time of infectiousness $\left(\langle\tau\rangle\right)$
- `R0`: Basic reproductive ratio (R_0)
- **infection-rate**: Rate of infection given contact with an infectious host (β)
- **end-infection-rate**: Rate of loss of infection $\left(\alpha \equiv \frac{1}{\langle\tau\rangle}\right)$
- **end-latency-rate**: Rate of loss of latency (γ)
- **infection-prob**: Probability of infection given contact with an infected host over time Δt (b). The formula relating the continuous infection rate β to the discrete infection probability b is given by
$$b \equiv 1 - e^{-\beta \Delta t} \tag{1}$$
- **end-infection-prob**: Probability of loss of infection over time Δt (a). The formula relating the continuous end infection rate α to the discrete end infection probability a is given by
$$a \equiv 1 - e^{-\alpha \Delta t} \tag{2}$$
- **end-latency-prob**: Probability of loss of latency over time Δt. This is derived from the continuous end latency rate γ using the formula
$$1 - e^{-\gamma \Delta t} \tag{3}$$

3 Numerical Simulation Parameters

- **model-time**: Denotes whether the model is a **Continuous** time model or a **Discrete** time model
- **time-step**: Time step used in discrete time models (Δt)
- **Discrete Approx**: Calculates discrete probabilites from continuous rates using equations (1)–(3)

4 Network Parameters

4.1 Network Creation

- **network-type**: Types of supported networks. See Section 5 for a descriptive list.
- **num-nodes**: Number of nodes (N)
- **lambda**: One of two possible parameters to vary properties of a specific network type. See Section 5 for its use in each type of network.
- **d**: One of two possible parameters to vary properties of a specific network type. See Section 5 for its use in each type of network.

4.2 Network Modification and Visualization

- **spawn-kill:** Specifies whether the user wants to *spawn* (create) or *kill* (remove) a node or link in the network. Used in conjunction with the **Node** and **Link** buttons.
- **Node:** When pressed, a node is created or destroyed (depending on selection of **spawn-kill**) when a user clicks in the world. If **spawn-kill** is set to **kill**, the nearest node to where the user clicks is destroyed.
- **Link:** When pressed, the user can create or destroy links (depending on selection of **spawn-kill**) as follows: on the first click of the mouse in the world, the closest node to this click is selected and set to white; on the second click of the mouse, a link is created or destroyed between the first selected node and the node that is closest in space to the second click.
- **Clear:** Clears the network, setting **num-nodes** to 0.
- **Randomize:** While pressed, performs random edge swaps between nodes without changing the degree sequence. Useful for exploring results of different network realizations of fixed degree sequences.
- **Metrics:** Prints network and simulation metrics. Current metrics include:
  - Simulation parameters
    - `<tau>`
    - `R0`
    - `Maximum number of simultaneous infections`
  - Network parameters
    - `Mean degree`: If each node's degree is denoted by k_i, then this is given by

$$\frac{1}{N}\sum_{i=1}^{N} k_i.$$

* `Edge density`: This is the number of links L over number of all possible links, given by
$$\frac{2L}{N(N-1)}.$$
* `Clustering coefficient`: If $N > 1$, computes the mean node clustering coefficient over all nodes. For a node i, let $\mathcal{N}_i$ denote the number of nodes that link to i. Let $tr(i)$ denote the number of links $\{j_1, j_2\}$ such that $j_1, j_2 \in \mathcal{N}_i$. The node clustering coefficient is thus given by
$$C(i) = \begin{cases} \frac{2tr(i)}{k_i(k_i-1)} & \text{if } k_i > 1, \\ \frac{2L}{N(N-1)} & \text{if } k_i \leq 1. \end{cases}$$
The clustering coefficient is thus given by
$$C = \frac{1}{N}\sum_{i=1}^{N} C(i).$$
* `Normalized clustering coefficient`: If $N > 1$, computes the mean normalized node clustering coefficient. The normalized node clustering coefficient is given by
$$C_{norm}(i) = \begin{cases} C(i)\frac{N(N-1)}{2L} & \text{if } L > 0, \\ 1 & \text{if } L = 0, \end{cases}$$
and thus the normalized clustering coefficient is given by
$$C_{norm} = \frac{1}{N}\sum_{i=1}^{N} C_{norm}(i).$$
* `Number of connected components`
* `Largest component (as proportion of network)`
* `Average path length in largest component`: If we denote the nodes in the largest component by $\{i_1, \ldots, i_m\}$, then the average path length in the largest component is given by
$$\frac{2}{m(m-1)}\sum_{j>k} l_{jk},$$
where l_{jk} is the number of edges in the shortest path betweek i_j and i_k.
* `Diameter of largest component`: The diameter of the largest component is the maximum shortest path in the largest component, i.e.
$$\max_{j>k} l_{jk},$$
where, as above, l_{jk} is the number of edges in the shortest path between two nodes i_j and i_k in the largest component.

- **Labels**: Toggles node labels on and off.
- **Scale**: Scales up the network to fill the world window. Useful in conjunction with **Spring**, which tends to contract everything down.
- **Spring**: Useful to "clean up" network. Simulates links as springs. While this is pressed and speed slider is set faster, the network will approach an equilibrium configuration. Particularly useful for trees (although fun to use with all network types).
- **plot-metric**: In conjunction with the **Network Metrics** plot window, selects the type of information to display. Current choices include:
  - **Degree Distribution**: Distribution of node degrees
  - **Clustering Coeffs**: Histogram of node clustering coefficients
  - **Normalized Coeffs**: Histogram of normalized node clustering coefficients
  - **Shortest Paths**: Histogram of shortest paths in largest component
  - **Probability Distribution**: Distribution corresponding to the **Custom Distribution** network type.
- **Update**: Updates the **Network Metrics** plot window to plot the metric chosen in **plot-metric**. Note that when loading a distribution from file, **plot-metric** is automatically set to **Probability Distribution** and the distribution is displayed. Also, any change in the currently displayed network that changes the number of nodes or links updates automatically the **Degree Distribution** plot.

5 Network Types

- **Complete graph**:
- **Empty graph**:
- **Erdos-Renyi**:
- **Nearest neighbor 1**: Consider changing the label (eventually). This gives $G_{RT}(N, d)$ aka $G_R^1(N, d)$ from text.
- **Nearest neighbor 2**: Current implementation of **Nearest neighbor 2** is as follows:
  - Change the interpretation of $d = 0$ so that it will give a grid $G_{RR}(n, n)$ without the diagonals instead of the empty graph.
  - Make creative use of input variable N (**number-of-nodes**) as follows:
    - Make the code compute the factorization $N = nm$ with $n \geq m$ that gives the largest possible integer value of m.
    - Then produce a grid with m rows and n columns.

    * For $N = 100$ and $d = 1$, this will give $G_{RD}(10, 10)$ as before.
    * For $N = 33$ and $d = 0$, this will give $R_{RR}(3, 11)$.
    * This would not allow us to produce, for example, $R_{RR}(2, 22)$, but we can get enough grids of various shapes to explore all interesting effects.

- **Small world 1**: Go with **Small World 1** as implemented. Consider changing the label (eventually). This gives $G^1_{SW}(N, d, \lambda)$ from text.
- **Small world 2**:
- **Preferential attachment**:
- **Generic scale-free**:
- **Spatially clustered**:
- **Random regular**:
- **Regular tree**:
- **Custom distribution**:

6 Setup & Go

- **New**: Creates a new network based on the **network-type** menu and corresponding network parameters **lambda** and **d**, if appropriate. *When using BehaviorSpace, the command **new-network** performs the same actions.*
- **Last**: Resets the node states to the previous initial conditions. *When using BehaviorSpace, the command **last-init** performs the same actions.*
- **Go**: Run the simulation. Press again to stop a simulation. *When using BehaviorSpace, the command **go** performs the same actions.*
- **Defaults**: Sets all simulation, disease and network parameters to defaults.
- **Load**: Loads information from file. The types of information that can be loaded include:
    - Degree sequence (see `degree.txt` for an example)
    - Degree distribution (see `distribution.txt` for an example)
    - Network (this is "Network Parameters" as referred to in this document, as well as nodes, links and node states – see `network.txt` for an example with a detailed format, and `network_simple.txt` for an example wit a simple format).
    - Parameters (this is "Simulation and Disease Parameters" as referred to in this document – see `parameters.txt` for an example).

  - All (Networks and Parameters) – see `all.txt` for an example).

- **Save**: Saves information to file. The types that can be saved include:
  - Network (this is "Network Parameters" as referred to in this document, as well as nodes, links and node states – see `network.txt` for an example).
  - Parameters (this is "Simulation and Disease Parameters" as referred to in this document – see `parameters.txt` for an example).
  - All (Networks and Parameters) – see `all.txt` for an example).

6.1 Initialize States

- **set-state-to**: Can be **Infectious** or **Removed**. Specifies the state you would like to initialize nodes to upon pressing the **Set** button.

- **set-state-by**: Specifies the way in which you would like to change the node states. Choice include:
  - **Number of nodes**: Needs to be an `integer`.
  - **Fraction of nodes**: Between 0 and 1.
  - **Vector from input:** Specify node labels in list format (e.g. **[0 1 10 16]**).
  - **Vector from file:** File should have one line with a list of node labels (e.g. **[0 1 10 16]**).

- **num/frac:** Stands for "number or fraction," depending on the choice of **set-state**. This is how you specify the number or fraction of nodes to infect or immunize.

- **min-deg:** Specifies the minimum degree for the population of nodes to apply the **set-state** algorithm to.

- **Set:** Applies the state initialization specified with all of the above choices. Note that this does not set all nodes to susceptible before applying the state changes. This is so you can, for example, say "Immunize 50% of the nodes and infect 10% of the nodes" by applying two different state changes in serial. The current implementation might be somewhat undesirable in that two serial state change operations might "undo" a state change from the first. A better implementation might be "Immunize 10 nodes, then infect 20 of the remaining" (or) "set the state of the nodes according to this probability distribution over the possible states." In fact, this might be a nice implementation: Remove the **state-type** selector and make it so that **num/frac** can store a list of 3 or 4 numbers representing the number or fraction of nodes in each state (S, E, I, R) or (E, I, R).

- **Reset:** Sets the state of all nodes to susceptible. If we implement what I suggested in the discussion on the **Set** button, then we can remove this button.

- **Select:** When pressed, a user can cycle through the states for a node by clicking on the node.
- **auto-set:** When set to **On**, every network created with the **New** button will automatically press the **Set** button.