**Technical Description of the Early Infection Outbreak Library**

Shah Jamal Alam

Version 1.1 (February 4, 2012) [in progress…]

1. **Introduction**

This library is developed in Java 1.6 using external Java libraries. The source code along with the documentation is available at the HIV Risk Dynamics project’s Google code repository[[1]](#footnote-1) under svn/trunk/episodicrisk-outbreaks/OutbreakCode.

* 1. **Setting up the code**

The project available at the Google code repository can be directly imported in the Eclipse IDE (or others). The code must be linked to the required additional open source Java libraries that are listed below.

***Additional libraries required:***

* JUNG 2.x library - http://jung.sourceforge.net/
* Flanagan Library – http://www.ee.ucl.ac.uk/~mflanaga/java/
* Apache’s Math Common - http://commons.apache.org/math/
* CERN Colt Library - http://acs.lbl.gov/software/colt/

The library is composed of two parts: generation of early infection outbreaks data and the summarizing population-level outputs and measures of the generated outbreaks size and shapes.

* 1. **Overview of the packages**

The library composes of six packages. An overview of each of the packages is given below (in alphabetical order):

* ***interfaces:*** This package contains the ParameterInterface interface that contains all constants and enum classes used by other classes in the library. All classes in the library must implement this interface. It also contains the interfaces for implementing the individual-based HIV transmission model main class, the class representing individuals in the model and the class storing data related to an HIV transmission event during a model run.
* ***cluster***: This package contains Java classes related to recording early infection outbreaks and infection trees. It also contains Java classes for outputting all and early infection transmissions recording during the given observed periods.
* ***display:*** This package contains classes for displaying the early infection forest as a JSwing component. The viewer and the renderer are based on the engines provided by the JUNG 2.x library.
* ***main****:* It contains two classes Controller.java and Summarizer.java. The Controller class is responsible for running the individual-based model by invoking the model main class and providing the input parameters that are read from a given input file (default: CSV). It collects population-level outputs from the model and generates output in the output directory in the project.
* ***plfit:*** The plfit package contains the wrapper class and the original code used by the Baek et al. (2011) for estimating the distribution parameters of a data that follows a heavy-tailed distribution. It also contains a wrapper class for accessing the C binary by Abramson et al. (2011) that fits a power-law distribution for a given data.
* ***reader:*** This package contains all the classes related to producing summaries of population-level and early infection outbreaks related statistics that are read from the generated data. All classes in this package implement the ParametersInterface in the *base-model* package. The classes in this package are called from the Summarizer class in the *main* package.

1. **Generation of early infection outbreaks data**

In this section, we use the individual-based model of Episodic Risk described in Zhang et al. (under review) to illustrate how to integrate the library code. We first describe the interfaces, followed by implementation of these interfaces and describing classes related to the Episodic Risk model and recording of the AHI outbreaks.

* 1. **Interfaces in the library**

The three Java interfaces are ParameterInterface, AgentInterface and the BaseModelInterface and are placed in the *basemodel* package.

* + 1. **ParametersInterface:** We start with the ParametersInterface, which contains all constant parameters and enum classes. This interface is implemented by all classes used in the library. Both the data generation classes and the summary outputs generation classes use this interface. The model related constants include the maximum number of iterations (simulation ticks), initial population of agents, proportion of initial infected individuals, duration of sexual activity, duration of acute and chronic stages of infection, base transmission probability etc, depending upon the features of the simulated model.

The Outbreaks related enum types are also stated in this interface:

* The OutbreakType enum gives the option of selecting the type of an early infection outbreak, which could be defined by the period spent in AHI (default), first six months of infected period and/or two years of infected period.
* The OutbreakRecord enum class defines the periods during simulation for which the early infection outbreaks are to be recorded. Currently, there is two such periods: Early Period and the Endemic. Users can change the start record time and the number of years for which the outbreaks to be recorded in the constructor of this enum class.
* The ChainsType enum defines the Types of transmission chains to be considered for measuring outbreaks’ statistics: continuous chains (ongoing transmission chains), dead-ends (dead transmission chains) or all chains.
* The NewChainThreshold constant defines the threshold that allows creating of entire new transmission chains during the recording period. This means that if a person transmits infection after this threshold, we consider the newly infected person as starting a new chain as a new root of an infection tree.

For details of the rest of the constants and the enum classes, refer to the JavaDocs of the library.

* + 1. **AgentsInterface:** This interface provides the methods that must be implemented by a class for an individual in the individual-based model. The outbreaks recorder class ClusterRecorder in the *cluster* package and other classes in the *cluster* package use these methods to retrieve an individual agent’s behavioral and infection states. Likewise, the Summarizer class and the associated classes in the *reader* package use the setter methods in the interface to reconstruct the infection forest and compute tree statistics.

For details on the methods in this interface, refer to the JavaDocs.

* + 1. **BaseModelInterface:** This interface must be implemented by any individual-based HIV transmission model. The model assumes a fixed time step schedule and is the main engine class responsible to connect all components of the IBM (e.g. sexual mixing, updating of agents’ status, and transmission events). An implementation of this interface must contain the outbreaks recorder ClusterEngine class as a member and implement the methods of this interface.

For details on the methods in this interface, refer to the JavaDocs.

* + 1. **TransmissionInterface:** The TransmnissionInterface ensures the basic information required to store a transmission event record. The toString() method outputs the transmission event to the ClusterEngine’s output recorder to output all transmission and early infection transmissions in a CSV file. Each transmission event is output as a row in the CSV file. One can extend this interface to account for more state variables about the infector and the infected agent in a more complicated model. See, e.g. the BaseTransmission class in the *cluster* package that implements this interface and the EpisodicRiskTransmission class in the *episodicriskmodel* package that extends theBaseTransmission class as an illustration.
  1. **Implementing the individual-based Episodic Risk model**

In this section, we illustrate how to use the library code using the Episodic Risk model (see the episodicriskmodel package). First, we implement the BaseModel interface in the EpisodicRiskModel class. This class is the main class responsible to contain collections for individuals, risk transitions, sexual mixing, birth/death and status update of the agents. It contains the member clusterRecorder, which is the instance of the ClusterRecorder class. The clusteRecorder is created in the setup function by calling the implemented function createClusterRecorder().

The model runs until the maximum iteration from the ParametersInterface. At each time step, the function run() is called, in which all processes in the model take place. In this run() method, we call the implemented function *callClusterRecorderStep()*. To add a transmission event in the clusterRecorder, we call the addTransmissionToClusterRecord(infector, susceptible) function inside infect(), where a susceptible agent is infected by a infector agent.

At each time step, the function updateIndividuals() is called inside the run() function in the model. Inside the updateIndividuals() function, the status of individuals is updated; also, birth and death processes take place. When an individual is dead (i.e. left the system), we call updateClusterRecord() to remove the dead individual’s record from the recorder.

The basic AgentInterface is extended to the EpisodicAgentInterface to account for additional states of an agent such that Risk-state and mixing-site. Likewise, other models can extend the AgentInterface for other behavioral/epidemiological states of an agent. The Person class then implements the EpisodicAgentInterface.

Finally, the BaseTransmission class in the *cluster* package implements the interface TransmissionInterface. We then extend the BaseTransmission class by the EpisodicRiskTransmission class (in the *episodicriskmodel*) package to extend the output about the transmission event such as the risk phase of the infector and the susceptible agent and the mixing site where the transmission occurred. Note that the EpisodicRiskTransmission overrides the toString from the BaseTransmission by first calling the base class’ toString() function and then appending further output.

Notice that in the createClusterRecorder() method (see above), we are required to pass the ‘transmission class’ as the last argument to create an instance of the cluster recorder at run-time. This ‘transmission class’ is either the BaseTransmission class (implementing the TransmissionInterface) or an extension such as the EpisodicRiskTransmission class.

* 1. **Implementing the individual-based Partnership model**

In this section, we illustrate how to use the library code using the Episodic Risk model (see the partnershipmodel package). First, we implement the BaseModel interface in the PartnershipModel class. This class is the main class responsible to contain collections for individuals, risk transitions, sexual mixing, birth/death and status update of the agents. It contains the member clusterRecorder, which is the instance of the ClusterRecorder class. The clusteRecorder is created in the setup function by calling the implemented function createClusterRecorder().

The model runs until the maximum iteration from the ParametersInterface. At each time step, the function run() is called, in which all processes in the model take place. In this run() method, we call the implemented function *callClusterRecorderStep()*. To add a transmission event in the clusterRecorder, we call the addTransmissionToClusterRecord(infector, susceptible) function inside infect(), where a susceptible agent is infected by a infector agent.

At each time step, the function updateIndividuals() is called inside the run() function in the model. Inside the updateIndividuals() function, the status of individuals is updated; also, birth and death processes take place. When an individual is dead (i.e. left the system), we call updateClusterRecord() to remove the dead individual’s record from the recorder.

The basic AgentInterface is extended to the PartnershipAgentInterface to account for additional states of an agent, similar to the EpisodicRiskAgentInterface (see Section 2.2). The PartnershipAgent class then implements the PartnershipAgentInterface.

We extend the BaseTransmission class (see Section 2.2) by the PartnershipTransmission class (in the *partnershipmodel*) package to extend the output about the transmission event. Note that the PartnershipTransmission class overrides the toString from the BaseTransmission by first calling the base class’ toString() function and then appending further output.

Notice that in the createClusterRecorder() method (see above), we are required to pass the ‘transmission class’ as the last argument to create an instance of the cluster recorder at run-time. This ‘transmission class’ is either the BaseTransmission class (implementing the TransmissionInterface) or an extension such as the PartnershipTransmission class.

* 1. **Implementing the individual-based Partnership model**

In the ParameterInterface, we have a list of directory paths and names of inputs/output file. The input file name is the CSV file, from which the running model controller reads a row (containing) the model parameters and passes on to the main model class. Data is generated into aggregates and individual runs. The controllers assume that a /data folder exist and subfolders /data/aggregates and /data/allruns. Please see the file paths in the ParameterInterface and modify, if required.

|  |  |  |  |
| --- | --- | --- | --- |
| **Implemented Model** | **Controller class in *main package*** | **Input file in /input folder** | **Status** |
| **Episodic Risk** | EpisodicRiskController | EpisodicRiskModelParams.csv | Available on GoogleCode |
| **Partnership model  (Jong-Hoon Kim’s)** | PartnershipModelController | PartnershipModelParams.csv | Available on GoogleCode |
| **Directional Sex model** | To be uploaded | To be uploaded | To be uploaded |

1. **Creating summaries of outputs from the generated data**

The Summarizer class in the *main* package reads the all transmissions and AHI transmissions for the existing multiple runs of a parameter set and then outputs summary statistics, which include:

* 1. **Population-level variables**

For each input parameter set, we record endemic prevalence and fraction of transmissions form acute (early) HIV infections in the simulated population during the observed period. In addition to that, the library generates joint distribution of AHI outbreaks with respect to size and duration categories. The default size and duration categories may be modified or extended. Other variable and their categories such as height can also be extended in the library.

* 1. **Early Infection Outbreaks**

We examine early infection outbreaks (clusters) from several aspects based on the outputs generated in a simulation run. The list is not exhaustive although it explores far more aspects of outbreak distributions than in previous simulation-based studies such as Murray (2002), Lewis et al. (2008) and Rocha et al. (2011); and phylogenetic studies such as Brenner et al. (2011).

In addition to the descriptive statistics for the summary variables (see ), we report further statistics to explore the distribution of the outbreak size distribution. For instance, the percentile measures tell us what would be the expected outbreak size (given an input parameters setting) in terms of the ranking of outbreaks, since the observed outbreak size distribution is positively skewed. The fraction of outbreak sizes indicates the density of the size distribution when the outbreaks are binned into the above categories with respect to size. For instance, a much higher proportion of ‘isolates’ indicate that under the given parameters setting, a big proportion of AHI outbreaks generated just a single transmission and were short-lived. On the other hand, a higher fraction of outbreaks of size greater than 10 indicates the AHI outbreaks’ role in carrying on the chains of transmission. Note that the above-mentioned categories were taken to be closer to the ones used by Brenner et al. (2011). Finally, we estimated the exponent (slope) of the outbreak size distribution from the information-theoretic approach used by Baek et al. (2011). In addition to that, we get the estimates for the largest outbreak size, which is similar to the actual maximum size of the outbreaks, under a given model setting.

Table 2: Summary of output variables and their description

|  |  |
| --- | --- |
| **Output summary** | **Description** |
| ***Early infection (AHI) outbreaks during the observed period*** |  |
| Size of all AHI outbreaks  (Average; Median; Max; Variance) | The size of an AHI outbreak is the total number of transmissions from individuals in that outbreak |
| Height of all AHI outbreaks  (Average; Median; Max; Variance) | The height of an AHI outbreak (tree) is the maximum distance (in terms of edges) from the root to a leaf. |
| Internal-to-leaf ratio of all AHI outbreaks  (Average; Median; Max; Variance) | (‘Internal’ is an individuals that transmitted at least once during acute stage. A ‘leaf’ is an individual that did not transmit during acute stage). This measure refers to the shape of the outbreaks |
| Duration of all AHI outbreaks  (Average; Median; Max; Variance) | Duration of an AHI outbreak is determined by the difference between the time (in days) of the last transmission and the first transmission from an individual in that outbreak. |
| Width of AHI outbreaks  (Average; Median; Max; Variance) | Width of an AHI outbreak is defined by the maximum children of a node in an outbreak tree (i.e. maximum number of secondary infections by an individual in an outbreak. |
| Height-to-Width ratio  (Average; Median; Max; Variance) | Ratio of height and width of an AHI outbreak |
| Empirical cumulative size distribution |  |
| ***Additional measures for outbreak size distribution*** |  |
| **Measure** | **Description** |
| Median size | Median or the 50th percentile of the AHI outbreak size distribution |
| 75th Percentile | 75th Percentile of the AHI outbreak size distribution |
| 90th Percentile | 90th Percentile of the AHI outbreak size distribution |
| 99th Percentile | 99th Percentile of the AHI outbreak size distribution |
| Fraction isolates | Fraction of outbreaks with just a single transmission (size = 1) |
| Fraction until 2 | Fraction of outbreaks with at most 2 transmissions (size ≤ 2) |
| Fraction until 5 | Fraction of outbreaks with size between 2 and 5 (2 < size ≤ 5) |
| Fraction until 10 | Fraction of outbreaks with size between 5 and 10 (5 < size ≤ 10) |
| Fraction greater 10 | Fraction of outbreaks with size greater than 10 (10 < size) |
| Inter-quartile mean | Average size of outbreaks taking into account of the inter-quartile range |
| Exponent fit of the slope | Exponent (slope) of the outbreak size distribution from Baek et al. (2011) |

* 1. **Patterns of Continuous transmissions**

We extend our exploration of the early infection (AHI) outbreak size distribution on their contribution in continuous chains of transmissions. We define continuous chains as the chains of transmissions that continue throughout the observed period during which early infection outbreaks were recorded. Those chains that end before that are termed as dead-ends.

For all the parameter sets, we get the cumulative distribution of the chronic infection transmissions (CHI) that link AHI outbreaks. A higher proportion of a larger number of CHI transmissions linking the AHI outbreaks could indicate smaller AHI outbreaks on the transmission chains whereas a higher proportion of CHI links of size 1 indicate larger AHI outbreaks on the chains.

1. http://code.google.com/p/hiv-risk-dynamics/ [↑](#footnote-ref-1)