FRED User's Guide

John Grefenstette, Shawn Brown, Jay DePasse, David Galloway, Bruce Lee, Yu-Ting Weng
University of Pittsburgh

Roni Rosenfeld, Alona Fyshe, Anuroop Sriram, Christopher Tischuk
Carnegie-Mellon University

Nathan Stone
Pittsburgh Supercomputing Center

Phil Cooley, Bill Wheaton
RTI International

26 Jan 2012

Acknowledgments

This work was supported by the National Institute of General Medical Sciences MIDAS grant 1U54GM088491-01. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Introduction

The accelerating growth in data availability and corresponding advances in high performance computing present new opportunities for *in silico* analysis of complex public health questions using computational modeling and simulation. FRED (A Framework for Reconstructing Epidemiological Dynamics) is an open source, modeling system developed by the University of Pittsburgh Public Health Dynamics Laboratory in collaboration with the Pittsburgh Supercomputing Center and the School of Computer Science at Carnegie Mellon University. FRED supports research on the dynamics of infectious disease epidemics, and the interacting effects of mitigation strategies, viral evolution, and personal health behavior. The system uses agent-based modeling based on census-based synthetic populations that capture the demographic and geographic distributions of the population, as well as detailed household, school, and workplace social networks. Multiple circulating and evolving strains can be simulated. Mitigation strategies in the framework include vaccination, anti-viral drugs, and school closure policies. FRED supports models of health behavior change to facilitate the study of critical personal health behaviors such as vaccine acceptance, personal hygiene and spontaneous social distancing. FRED is available through open source in the hopes of making large-scale agent-based epidemic models more useful to the policy-making community, the research community, and as a teaching tool for students in public health.

Overview of this document

FRED is a research tool and this document is aimed at a target audience of infectious disease modelers with a high degree of computational experience. The pragmatics of obtaining, installing and running FRED have been described in the FRED Quick Start Guide. Please refer to that document for instructions, and some quick hands-on exercises. This document will describe FRED on a conceptual level. Developers will also want to consult the code-level documentation and even the code itself for specific implementation details.

Synthetic Population

FRED uses the synthetic population developed by RTI, International. In summary, RTI used a proportional iterative method developed in (Beckman, et al. 1996) to generate an agent population from the US Census Bureau's Public Use Microdata files (PUMs) and Census aggregated data. See (Wheaton, et al. 2009) for a detailed description. Each agent had a set of socio-demographic characteristics and daily behaviors that included age, sex, employment status, occupation, and household location and membership. (This section is based on the Supplemental Materials from (Cooley et al, 2011).

Assignment of students to Schools

The synthetic population also represented schools and assigned persons of school age to schools using methods described in (Cajka et al., 2010). Using information from the National Center for Education Statistics (NCES) a database of all public and private schools in the US was developed, including each schools geolocation, and age-specific capacity. A set of heuristics were developed to assign each school-age child to a specific age-appropriate schools, using several assumptions including:

- Geographic proximity is a major criterion for making assignments.
- Students are assigned to a school on the basis of distance along a network (roads) rather than distance along a straight line.

- Students attend school only in their county of residence.
- Students are assigned to a school according to the school's capacity for their grade.
- No special allowances are made to assign siblings to the same school, other than the fact that they shared the same geographic location and therefore should be assigned to the closest school that had capacity for their grade levels.

Workplace Data and Allocation Model

The RTI synthetic population also assigned employees to workplaces, taking into account:

- the number of persons who lived in one Census tract but worked in another and
- the number of workplaces by size by the same Census tract.

Based on the number of firms by firm size category and Census block group, synthetic workplaces were created and located at the centroid of the block group indicated by the workplace's address. Workers were then assigned to those workplaces so that workers who reported working in a specific block group were assigned at random to a firm located within that block group. The workplaces also included schools, hospitals and other types of institutions that could be used to specifically track special synthetic agents such as teachers, health care workers, and others.

One important issue in the STP64 data is how the Census asks the question that is the source of the commuting estimate. Respondents were asked to identify the place they spent the most time working at *in the previous week*. This means that the US dataset contains data on regular commutes to the individual's typical workplace as well as occasional work-related trips. As work trips lasting most of a week can be expected to involve longer distances than a typical commute, one might attribute the greater than expected number of very long distance commutes to such occasional work-related travel.

Agent Model

Agents in FRED represent individuals in a population living in a specific geographic region. For example, many of the initial studies with FRED model the population of Allegheny County surrounding Pittsburgh, Pennsylvania. The model of the population of Allegheny County includes 1,242,755 agents. Each agent has associated with its demographic information (e.g., age, date-of-birth, sex), health information (e.g., current health status, date of infection, level of symptoms, infectiousness, susceptibility), locations for social activity (household, neighborhood, and possibly school or workplaces), and health-related behaviors (e.g., probability of getting a vaccine or staying home when sick). During each simulated day, agents interact with the other agents who share the same social activity locations. If an infectious agent interacts with a susceptible agent, there is a possibility of transmitting a disease from the infected agent to the susceptible agent. FRED simulates the population of agents during a period of time, usually several months, and tracks the spread of disease among the population. Since each infection event is recorded, it is possible to analyze the course of an infection through the population, and to evaluate several possible control measures.

The main program is contained in **Fred.cc.** For further details about the representation of individual agents, see files: **Person.cc, Population.cc**.

Demographics

The population input file specifies the sex, the age (in years), the marital status, and the profession of each agent. By default, these demographic features remain constant during a simulation run. FRED agents are given a random birthday such that the age in years agrees with the age listed in the population input file.

FRED also supports dynamic demographics: aging, births and deaths. These optional features are controlled by the parameters **`enable_aging`**, **`enable_births`**, and **`enable_deaths`**, respectively. If aging is enabled, then an agent's age increases on each birthday. If births are enabled, then each female of child-bearing age may become pregnant on any simulation day using age-specific maternity rates provided in the yearly_maternity_rate_file. Upon becoming pregnant, the agent is assigned a due-date based on a Gaussian distribution with a mean of 280 days and a standard deviation of 7 days. When the due-date arrives, the mother gives birth to a new agent who is assigned a random sex and is assigned to the same household as the mother. If death is enabled, then on each agent's birthday, it is decided whether that agent will die during the coming year using age-specific mortality rates provided in the yearly_mortality_rate_file. When an agent dies, it is removed from the population. For further details, see the source file **Demographics.cc**.

Health

Each agent maintains a list of current infections (one for each disease). An infection follows a natural history (e.g., latent period, infectious period, symptomatic period, as well as infectiousness) as specified via input parameters. Agents' health information also includes disease status (S, E, I, R), immunity, at-risk status, susceptibility, current symptom levels, and how many others have been infected by this agent. See **Health.cc** for further details.

Activities

An agent follows a daily pattern of interactions with groups of other agents. All interactions in FRED occur in a specific place. The types of places in FRED include: Households, Neighborhoods, School, Classrooms, Workplaces and Offices. Classrooms are small mixing groups with a given School. Offices are smaller mixing groups with Workplaces. Each agent maintains list of "favorite places", at most one for each of the above types. Agent may have undefined favorite places if they do not participate in that activity. For example, adults do not have a defined School or Classroom. On a typical day, the agent interacts with other agents in each favorite place that is defined.

Neighborhoods are defined on a grid with 1 km square cells. The agent's home neighborhood is the cell in which its household is located. However, an agent may visit another neighborhood in the community during a given day. The decision about where to spend the neighborhood activity period is made independently each day, with the highest probability to visit the home neighborhood, and a lesser probability to visit one of the surrounding neighborhoods, and a small probability of visiting a randomly selected neighborhood within a given community radius.

If an agent is infectious, then any location the agent visits during that day is considered an infectious location. Susceptible agents can only become infected at an infectious location, so interactions among agents at non-infectious locations need not be simulated.

Schools are closed on weekends and during schedule summer holidays. Schools may also be closed due to school closure policies. Students do not visit their school when the school is closed.

Similarly, most workers do not visit their workplaces on weekends. However, some workers are designated as weekend workers, and they continue to visit workplaces on weekends.

To reflect weekend schedules of schools and workplaces, the number of neighborhood contacts is increased by 50% on weekends.

See **Activities.cc** for more details.

Behaviors

FRED is designed to include any number of health-related behaviors. New behaviors can be added with minimal programming effort. Each behavior involves a decision on the willingness of the agent to perform the behavior. The current set of behaviors includes:

1. **Stay home when sick**: If an adult is symptomatic, is that person willing to stay home? If so, the agent withdraws to the household, does not interact with other in the neighborhood, at work or at school. The agent also does not begin new overnight travel.
2. **Keep child home when sick**: If a child is symptomatic, is the child's adult decision-maker willing to have the child withdraw to the household. In this case, the same restrictions on contact apply as in the adult "stay at home when sick" behavior.
3. **Accept vaccine**: Is an adult willing to accept a vaccine, if one is available?
4. **Accept vaccine for child:** This is the adult's willingness to have a child vaccinated.
5. **Accept another vaccine dose**: If a vaccine requires more than one dose, is the agent willing to accept an additional dose?
6. **Accept another vaccine does for child:** Same as above, but the decision is made by an adult on behalf of a child.

Future behaviors may include: wearing a face mask; taking anti-viral prophylaxis; staying home when well; keeping children home when well; avoiding travel; avoiding neighborhood contacts; hand-washing; and others.

Adult Decision-makers for Children

The synthetic population used by FRED was developed by RTI, International using a process described in (Wheaton et al, 2009). Household level Public Use Microdata Sample (PUMS) includes the ages, sexes, and relationships of all individuals within a household. Households from the PUMS are selected with replacement to make up the synthetic population, selecting households so that the census counts and other statistical distributions are maintained at the census block level. As a result of this process, the FRED synthetic population includes information giving the relationship of each member of the household to the Householder (typically, the owner of the house or the head of the household.) This information is used to assign an adult in the household as the responsible decision-maker for the health-related behaviors of each child in the household. The rules for selecting the adult decision-maker for each child are as follows:

1. If the Householder is the parent (natural parent, adoptive parent, step-parent) of the child, then the Householder is designated the child's decision-maker.
2. If the Householder is the grandparent of the child and there is an adult in the householder who is a child of the Householder, then the first such adult is designated as the decision-maker for the child. Note that the household relationship data does not provide enough information to determine whether such an adult is actually the parent of the child in question.
3. If the Householder is the grandparent of the child and not plausible adult parent is present in the household, then the Householder is designated the decision-maker for the child.
4. Otherwise, a random adult in the household is designated as the decision-maker for the child.

The rules above permit multiple decision-makers per household. No preference is made on the basis of sex or age, other than that each decision-maker is an adult (i.e., at least 18 years old.)

Behavior Strategies

How real people make health decisions is an active area of research without an obvious consensus theory. Indeed, it seems likely that different people use different methods to come to decisions about health-related behavior. FRED agents can apply a variety of strategies to determine their willingness to adopt a given behavior. Each agent may revisit its willingness to perform the give behavior. Thus each strategy specification includes a frequency parameter that determines how often agents make decisions about their willingness to perform the behavior.

1. **Refuse**: Agent is never willing to perform the given behavior.
2. **Accept**: Agent is always willing to perform the given behavior.
3. **Flip Behavio**r: Agent is assigned a fixed probability *p* of being willing to perform the given behavior. The agent revisits its willingness to perform the behavior according to the frequency parameter.
4. **Imitate Prevalence:** The agent is assigned an initial probability *p* of being willing to perform the given behavior. The agent revisits its willingness to perform the behavior according to the frequency parameter. When reconsidering the decision, the agent estimates the prevalence of willingness among the agents in its social networks: household, neighborhood, school and workplace. The estimate is a weighted average of the actual prevalence in each group. Given the weighted estimate, the agent adjusts its probability *p* toward the perceived prevalence. For example, if the agent perceives that the prevalence of willingness is 0.75, then it adjusts its own probability to be closer to 0.75.
5. **Imitate Consensus:** This strategy is similar to the Imitate Prevalence except that if the weighted estimate of prevalence exceeds a threshold, the agent adjusts its probability *p* toward 1; otherwise the agent adjusts its probability toward 0. For example, if the agent's threshold is 0.5, then if the agent perceives that the majority of its associates is willing to perform the behavior then the agent becomes more likely to accept the behavior; otherwise the agent becomes more likely to refuse the behavior.
6. **Imitate by Count:** This strategy is similar to the Imitate Consensus except that if the weighted number of nearby agents exceeds a threshold, the agent adjusts its probability *p* toward 1; otherwise the agents adjusts its probability toward 0. For example, if the agent's threshold is 3.0, then if the agent perceives that at least three its associates is willing to perform the behavior then the agent becomes more likely to accept the behavior; otherwise the agent becomes more likely to refuse the behavior.
7. **Health Belief Model:** According the Health Belief Model, people make health behavior decision based on several specific considerations: susceptibility, severity, benefits, and barriers.
    a. Perceived Susceptibility refers to the person estimate of how likely he or she is to become adversely affected by the disease or condition;
    b. Perceived Severity refers to the level of adverse consequences that are likely if the person become affected;
    c. Perceived Benefits refers to the estimated protective effects of the behavior; and
    d. Perceived Barriers refers to the conditions that may prevent the agent from perform the behavior.

These constructs are clearly specific to the particular health behavior under consideration, so including an HBM strategy for a particular behavior in FRED requires customized programming. However, once the perceptions are computed, they can be combined into a decision rule using formulas developed by David Durham (Durham, 2010). These formulas have been implemented in the FRED Behavior module, and are controlled by run-time parameters. (See Parameters Section.)

Each agent is assigned a strategy independently for each behavior defined for that agent.

Population-Level Market Shares

For each behavior in FRED, the user may specify the fraction of the population using each strategy for that behavior. For example, it might be desirable to investigate the effect of varying the fraction of the population using the **Accept**, **Refuse**, and **Imitate Consensus** strategies. The user can specify a given distribution, for example, 20% of the population adopts the **Accept** strategy, 30% adopts the **Refuse** strategy, and 50% adopt the **Imitate Consensus** strategy for a given behavior. The share of the population can be specified separately for each behavior.

See files **Behavior.cc, Health_Belief_Model.cc** and **Random_Behavior_Model.cc** for further details.

Place Model

All infections in FRED are transmitted from one agent to another in some particular place. The types of places in FRED include: Households, Neighborhoods, School, Classrooms, Workplaces and Offices.

The locations input file specifies the households, school and workplaces in the modeled region. The household for each agent in the population file is required to be in the locations file. The schools and workplaces are optional (since an agent may attend a school or workplace beyond the limits of the modeled region.)

Neighborhoods are defined on a grid with 1 km square cells. The agent's home neighborhood is the cell in which its household is located. However, an agent may visit another neighborhood in the community during a given day. (See Agent Model section.)

Classrooms are small mixing groups with a given school. Classrooms are defined by dividing up all the students who attend a given school into separate age groups. Each age group is divided into classroom groups of up to 40 students. A student interacts with the students assigned to the same classroom for the entire school year. A student also interacts (with a separate probability) with all the students attending the same school.

School and classrooms are closed on weekends, during scheduled summer breaks, and possibly due to school closure policies.

Offices are small mixing groups with a given workplace. Offices are defined by dividing up all the workers in a given workplace groups of up to 50 workers. A worker interacts with the other workers in the same office, and, with a separate rate, with all workers in the same workplace.

For further details, see files: **Place_List.cc, Place.cc, Household.cc, Neighborhood.cc, School.cc, Classroom.cc, Workplace.cc, Office.cc**

Disease Model

FRED supports multiple diseases circulating in the same population. Each Disease has separate parameters specifying transmissibility, mortality rate, its natural history (e.g., latent period, infectious period, symptomatic period). See **Disease.cc**

Each disease has an associated Epidemic object that keeps track of population level statistics associated with the disease, such as the number of agents that are Susceptible, Exposed,

Infectious and Recovered. The Epidemic object prints out the daily reports to the output file. See **Epidemic.cc**.

Transmission Model

The core phenomenon of an epidemic in FRED is the spread of an infection from one agent to another in a particular place. Each type of place represents a distinct environment for the spread of infection. Each type of place is characterized by two sets of numeric parameters:

- the number of contacts per infectious person per day, and
- the probability that a contact transmits an infection

The number of contacts per day for each type of place is a tunable parameter, and is set through the process described in the Calibration Section.

The transmission probability for a give place type generally depends on the age of the infectious person and the susceptible person. These are specified as vector input parameters.

The key method implementing infection is called Place::spread_infection() in **Place.cc**. This method is called once a day for each infectious place (having at least one infectious visitor). The method follows the following procedure:

For each infectious person i, the expected number of contacts is:

```
Num_contacts(i) = Trans(D) * CR(P) * Inf(i) * S(P)/N(P)
where  Trans(D) is the transmissibility factor for disease D,
       CR(P) is the contact rate for place P,
       Inf(i) is the infectivity of agent i,
       S(P) is the number of susceptible agents visiting place P, and
       N(P) is the number of total agents who usually visit place P

       For contact number 1..Num_contacts(i)
              pick a susceptible agent j from today's visitors;
              let PROB = Trans_prob(i.j) * Sus(j)

              where   Trans_prob(i.j) is the transmission probability from i to j, and
                      Sus(j) is the susceptibility of agent j.

              If a random number R is less than PROB, then agent i infects agent j.
```

For further details, see **Place.cc, Household.cc**

Pandemic influenza model parameterization

FRED is parameterized for a default pandemic influenza strain following the process described in (Cooley P, Brown S, Cajka J, Chasteen B, Ganapathi L, Grefenstette J, Hollingsworth CR, Lee BY, Levine B, Wheaton WD, Wagener DK. The Role of Subway Travel in an Influenza Epidemic: A New York City Simulation. J Urban Health. 2011 Aug 9. [Epub ahead of print] PubMed PMID: 21826584.) Paraphrasing the Supplementary Material from (Cooley et al, 2011):

The pandemic was assumed to have the age-dependent attack rate pattern of the historical 1957-8 "Asian" influenza A (H2N2), see Longini et al. Accordingly, we calibrated our model using the Ferguson et al. approach from historical (1957–58, 1968–69) influenza pandemics. We specifically used the 30–70 rule developed by Ferguson et al. in which 70% of all transmission occurred outside the household: 33% in the general community and 37% in schools and workplaces.

Following (Cooley et al, 2011), we adopted that additional requirement that transmission rates in schools are double those in workplaces. Calibrating the model involved targeting an epidemic with a 33% attack rate (AR) consistent with the age specific parameters derived from the 1957-58 pandemic. Daily contact rates were treated as endogenous parameters and were interpreted as the daily contact rates that reproduced a pandemic with a 33% AR in a population with no acquired immunity and satisfied the 30–70 rule. Therefore, our estimated contact patterns produced an epidemic designed to be similar in transmissibility to the 1957–58 epidemic with an AR of 33% and a basic reproductive rate (R0) of approximately 1.4.

The calibration process using the Allegheny County synthetic population results in the following default parameters:

```
neighborhood_contacts[0] = 42.32
school_contacts[0] = 15.83
workplace_contacts[0] = 1.66
classroom_contacts[0] = 31.66
office_contacts[0] = 3.32
```

As in (Cooley et al, 2011) we assumed that 50% of sick individual stay at home and do not interact with anyone outside of the household. Note that our default school absentee rate is generally lower than other models (e.g., Ferguson et al. use a 90% absentee rate). Additionally, we assumed that all community contacts increase by 50% on weekends.

Contacts within Household

Calibration to the 30-70 target criteria was impossible unless within household contacts were treated differently than other locations. Following (Cooley at al, 2011), we assumed that each pair of agents within a household make contact each day with a specified probability. This probability is tunes as part of the calibration step to achieve the 30-70 target distribution. The rersulting contact probability for Allegheny County is:

```
household_contacts[0] = 0.19
```

Interventions

Vaccines

FRED provides a fairly robust capability for simulating the use of vaccines during a pandemic. Multiple vaccines can be simulated simultaneously, with differing administration schedules and target groups, and with different efficacies. Each vaccine can also have multiple doses and be restricted by age. It is also possible to model varied vaccines schedules by day. Prioritization by age groups, or by ACIP recommendation is available with the capability to vaccinate only the priority group. Currently, vaccines can only be applied to one disease.

Vaccines in FRED are currently modeled as so-called "all or nothing" vaccines. Each vaccine is given an age-specific efficacy and efficacy delay. When an agent takes a vaccine, there is a random draw to determine whether the vaccine will be efficacious for that agent. If it is not, then the vaccine has no effect until another vaccine or dose is administered. If the vaccine dose is efficacious, then the agent will become immune to the disease after the specified efficacy delay. As in real life, the agent has no knowledge as to whether their dose of vaccine was efficacious, and so if they are exposed after a failed vaccine or during the delay period, they may get sick from the disease.

Vaccination programs currently implemented can be thought of as mass vaccination strategies. At the beginning of the simulation, a set of queues is set up based on prioritization of the agents. These queues are then randomized and as vaccines are put into the system, agents can choose

whether or not to accept a vaccine. To determine this decision, the simulation can use a straight coverage probability, or a more complex behavioral model. Heads of households can make decisions for younger members.

Antiviral drugs

School closure

FRED includes two school closure policies: global and individual. There are two triggers for the global school closure policy (**`school_closure_policy = global`**). First, all schools decide to close on the simulation day specified by the parameter **`school_closure_day`**, unless that parameter is negative. Second, all schools decide to close if the population attack rate exceeds a threshold (**`school_closure_threshold`**). With either trigger, school closure is delayed by a number of days indicated by parameter **`school_closure_delay`**. Schools reopen after a number of days indicated by parameter **`school_closure_period`**.

If the individual school closure policy is selected (**`school_closure_policy = individual`**), then each school is closed if the attack rate within the school exceeds a threshold (**`school_closure_threshold`**). School closure is delayed by a number of days indicated by parameter **`school_closure_delay`**. Schools reopen after a number of days indicated by parameter **`school_closure_period`**, but may close again if the school attack rate exceeds the threshold.

The default is no school closure policy: **`school_closure_policy = none`**

School are always closed on weekends. All schools also close for the summer if the parameter **`school_summer_schedule`** is set. In that case, schools are closed between the dates specified by parameters **`school_summer_start`** and **`school_summer_end`**, inclusive.

For details, see **School.cc.**

Future Interventions

It is planned to include other interventions in FRED, including: quarantine; travel restrictions; environmental interventions (e.g., vector control); behavioral changes; official announcements and warnings; and education campaigns.

Geography and Travel

FRED represents geography as a hierarchy of fixed square grids. Currently there are three layers in the hierarchy, called Large Grid, Grid, and Small Grid.

The Large Grid consists of 20km x 20km cells by default. The Large Grid is aligned global geo-coordinate system. Cells in the Large Grid store the population count for the cell, along with a vector of all persons residing in that cell. In the future, the Large Grid may be appropriate for storing climate or other environmental profiles.

The medium grid, called Grid, consists of 1km x 1km cells by default. These cells function as neighborhood units, and store information about the preferred schools and workplaces attended by people living with the cells. This information is used when agents need to change schools, or leave school and start to work.

The Small Grid consists of 10m x 10m cells. In the future, these cells will store fine-resolution information like the number of mosquitoes and the presence of water containers.

For further details, see source files: **Large_Grid.cc, Large_Cell.cc, Grid.cc, Cell.cc, Small_Grid.cc, Small_Cell.cc**

As an optional feature, agents can travel overnight for one or more days. When an agent is on overnight travel, the traveling agent (called the "visitor") resides in the household associated with another agent (called the "visited agent"). The visitor interacts with agents in the visited agent's household and neighborhood. If the visitor is employed, then the visitor also interacts with the visited agent's office and workplace. Visitors do not attend school while traveling.

If travel is enabled, two additional input files are required, a cell population file and a trip list file. The cell population file contains the total population for each cell, considering the entire U.S. population. The trip list file contains a large sample of trips from one cell to another. This file covers the entire U.S. The trip file can contain samples based on data obtained from air travel databases or from any other source considered appropriate. The current default is a sample of 5 million cell-to-cell trips based on a gravity model of travel, using the formula:

$$\text{Prob\_travel}(i,j) = \text{Pop}(i) * \text{Pop}(j) / (K * \text{Distance}(i,j))$$

where Pop(i) is the total population residing in cell i (derived from the entire U.S. synthetic population), Distance(i,j) is the distance in kilometers between the center of cells i and j, and K is a normalization factor so that Prob_travel(i,j) sums to 1.0. Given the pdf defined by the above formula, we select 5 million trips (with replacement) and store these in the trip list file.

During the Travel::setup() method, the trip list sample is read, and those trips involving the model region are retained. The number of daily trips involving the population in the model region is proportional to the fraction of the retained trips. The goal is that the expected number of overnight trips involving agents in the modeled regions remains invariant, regardless of the size of the model region.

The cell population file is used to set the probability of initiating a trip between two cells. If a trip between cells i and j is selected, it is only initiated with probability dens(i)*dens(j), where dens(i) is the fraction of the total population in cell i that actually occurs in the synthetic population for the current model region. For example, if cell i is on the border of the model region and happens to contains only 50% of the entire cell population according to the current model population, then the probability of any trip to or from cell i is reduced by 50%.

Run-time Parameters

The run-time parameters for FRED are contained in two parameter files. The first file is **$FRED_HOME/input_files/params.default** and contains the default values of all defined run-time parameters. **This file should not be modified.** The second file is usually called **params** and contains any parameter values that override the default values. The **params** file may be empty.

Both files have the same format. Lines that begin with a "#" character are considered comments and are ignored. Parameters with scalar values are specified with lines of the form:

```
<name> = <value>
```

For example:

```
days = 100
diseases = 1
popfile = pop_Alleg.txt
```

Some parameters are vector valued, in which case the format is:

```
<name> = <size> v_1 v_2 ... v_size
```

For example:

```
# cdf of trip duration in days
travel_duration = 6 0 0.2 0.4 0.6 0.8 1.0
```

If a parameter appears more than once in a parameter file, the last setting takes precedence. If a parameter appears in both **params.default** and **params**, the value in **params** overrides the value in **params.default.**

Input File Parameters

FRED requires input files to describe the population of agents and the locations they visit. Another input file specifies the number of new cases that are seeded into the population on each simulation day.

There are several other optional input files that are required only if certain features are enabled. The following table describes the input file formats and related parameters.

Table 5.1: Input File Parameters

Parameter	Type	Definition, Defaults and Notes
`popfile`	string	Required file containing one line per person. **Default:** $FRED_HOME/region/loc_Alleg.txt Format: **ID AGE SEX MARRIED OCCUPATION HOUSEHOLD SCHOOL WORKPLACE RELATIONSHIP** **Note:** Since these files are usually large, you may want to store them in a centralized location
`locfile`	string	Required file containing one line per location. **Default:** $FRED_HOME/region/pop_Alleg.txt Format: **ID TYPE LAT LON** where ID is a unique string; TYPE is one of {H, S, W, M} for Household, School, Workplace or Hospital, resp; and LAT, LON is the latitude and longitude **Note:** Since these files are usually large, you may want to store them in a centralized location
`cell_popfile`	string	Optional file with the initial population per 20km-grid cell. **Default:** $FRED_HOME/region/cell_pop.txt Format:

		COL ROW POPULATION **Note:** Only used if enable_travel = 1
`tripfile`	string	Optional file containing sample of trips between 20km-grid cells. **Default:** $FRED_HOME/region/trips.txt Format: **SRC_COL SRC_ROW DEST_COL_DEST_ROW** **Note:** Only used if enable_travel = 1
`primary_cases_file[d]`	string	Required files giving the number of primary infections to introduce for each simulation day. **Default:** $FRED_HOME/input_files/primary_cases_schedule_0.txt (for disease 0) FORMAT: time step map (see text).
`vaccination_capacity_file`	string	Optional file giving vaccine availability. **Default:** $FRED_HOME/input_files/vaccination_capacity-0.txt (for vaccine 0) Format: Time step map: START_DAY NUMBER_OF_DOSES **Note:** The number of doses is added to the system capacity every day until the day given on the following line, or until the end of the simulation.
`yearly_birth_rate_file`	string	Optional file containing age-specific birth rates for females. **Default:** $FRED_HOME/input_files/birth_rate.txt Format: **AGE BIRTH_RATE** where the rate is the probability of giving birth at the given age in years. **Note:** Only used if enable_births = 1
`yearly_mortality_rate_file`	string	Optional file containing age-related mortality rates. **Default:** $FRED_HOME/input_files/ortality_rate.txt Format: **AGE M_RATE F_RATE** where the rates are for males and females of the given age in years. **Note:** Only used if enable_deaths = 1

Population file format

The population file containing one line per agent, and each line has the following format, where fields are separating by white space:

```
ID AGE SEX MAR OCC HOUSEHOLD SCHOOL WORK RELATION
```

Description of fields:

FIELD **TYPE** **MEANING**

ID	STRING	unique agent id
AGE	INTEGER	agent age in years
SEX	CHAR ('M', 'F' or 'U')	sex of agent
MAR	INTEGER	marital status
OCC	INTEGER	occupation code (not currently used)
HOUSEHOLD	STRING	id of agent's household
SCHOOL	STRING	id of agent's school
WORK	STRING	id of agent's workplace
RELATION	INTEGER	relation to the head of household

All fields are required. The location IDs indicate the agent's "favorite places", and correspond to IDs in the location file. The ID value -1 indicates that a location type does not apply to the agent.

RELATION represents the relationship between the householder and the individual. The possible values are

1 . Householder
2 . Husband/wife
3 . Natural born son/daughter
4 . Adopted son/daughter
5 . Stepson/stepdaughter
6 . Brother/sister
7 . Father/mother
8 . Grandchild
9 . Parent-in-law
10 . Son-in-law/daughter-in-law
11 . Other relative
12 . Brother-in-law/sister-in-law
13 . Nephew/niece
14 . Grandparent
15 . Uncle/aunt
16 . Cousin
17 . Roomer/boarder
18 . Housemate/roommate
19 . Unmarried partner
20 . Foster child

Primary Cases File Format

The primary cases file is a required file giving the number of primary infections to introduce for each simulation day. The default format is:

```
#line_format
# the default 100 seeds on day zero...
0 0 100
```

The full format is:

```
start end attempts [ strain [ prob [ min [ lat lon radius ] ] ] ]
```

Only the first three fields are mandatory. The first two fields give the starting and ending day, and the third field specifies the number of attempted infections per day. For each specified day we attempt to generate new cases by randomly selecting agents (with replacement) and infecting them if they are susceptible. Note that the actual number of infections may be less than the

number of attempts because some selected agents may already be infected or may be immune. The process continues until the end day indicated on the same line in the file, or until the end of the simulation.

The fields others are optional, but if present, must be given in the order above. If a location is specified, then all three location fields must be present (lat, lon & radius). The radius is specified in kilometers. To disable geographic seeding either omit lat, lon & radius or give a value for radius that is greater than 40075 or less than zero.

The example below will make 100 seeding attempts of strain 0 on day 0, each with attempt probability of 1, requiring a minimum of 100 transmissions, all selected randomly from people whose households are within 100km of the specified point.

0 0 100 0 1 100 40.44181 -80.01278 100

Output Parameters

FRED produces several output files. The level of detail can be control by parameters described in the following table.

Table 2: Output Parameters

Parameter	Type	Definition and Notes
outdir	string	Directory containing the output files. If the string beings with "/" it is interpreted as an absolute path. Otherwise, it is relative to the current working directory. Default: OUT
verbose	int	If set, print information for monitoring system progress to the standard output. Higher values produce more output. Default: 1
debug	int	If set, print verbose debugging output to stdout. Higher values produce more output. Default: 0
track_infection_events	int	If set, then a file called infections<n>.txt is created for run <n>. This file contains one line per disease transmission event, showing the id of the infector, the infectee, and various other information. The format for the infections file is: DAY DISEASE_ID HOST_ID HOST_AGE INFECTOR_ID INFECTOR_AGE PLACE_ID If track_infection_events > 1, additional data is written on each line. For further details, see: **Infection.cc.** Default: 1
quality_control	int	if set, information about the size and age distribution for the various types of places is printed out the Log file. Default: 1
rr_delay	int	Identifies the number of days between the definition of a cohort and the reporting of that cohort's reproductive rate in the output file. See examples below. Default: 20
output_population	int	If set, a file containing the current population will be output periodically. See explanation below. Default: 0
output_population_date_match	string	If output_population is set, dump the population

		on any date that matches this string. The format is DD-MM-YY, with * matching any value. Default: = 01-01-*
`pop_outfile`	string	Name of population dump file. Default: pop_out

Output file format

The outfile (called out<n>.txt for run n) contains one line for each simulation day of the run. The format of the file is:

Day:	Current day counter
Str:	Disease id
S:	Number of agents in Susceptible state for this disease
E:	Number of agents in Exposed state
I:	Number of agents in Infectious state
I_s:	Number of Infectious agents who are symptomatic
R:	Number of agents in Removed (Recovered) state
M:	Number of agents that are Immune
C:	Number of current Cases (new E's)
N:	Population size
AR:	Attack Rate
CI:	Number of new symptomatic cases
CAR:	Clinical attack rate
RR:	Reproductive rate
NR:	Number in the cohort used to compute RR
Day_of_week:	Current day of week, e.g., `Wed`
Date:	Calendar date associated with the simulation day, eg, `2011-01-05`
Year:	Epidemiological year
Week:	Epidemiological week (`1-53`)

Note: RR is the reproductive rate observed for a cohort of individuals who were exposed on the same day. NR is the size of the cohort. The day for which the cohort is defined in given by the parameter `rr_delay`. The default is:

```
rr_delay = 20
```

This value means that on day 20 of the output file, the RR for the cohort exposed on day 0 is reported. On day 21, the cohort exposed on day 1 is printed and so on. The delay should be made long enough to capture all the infectees of the cohort.

Periodic Population Dumps

If the parameter `output_population = 1`, then a file will be written on the start day, the end day, and on any day matching `output_population_date_match` parameter. The file will be a dump of the population that will be identical to the input population file, but will have additional fields for the classroom and office ids (which are both set at runtime).

Global Control Parameters

The following parameters provide basic control of FRED simulations.

start_date: the calender date corresponding the simulation day 0. Format YYYY-MM-DD.

```
start_date = 2011-01-01
```

days: the number of days in a single simulation run. FRED runs for the given number of days regardless of the epidemic state (that is, FRED does not stop early if no one is currently infected.)

```
days = 120
```

seed: the seed for the random number generator. The seed values for all runs of the simulation are based on the initial seed and the run number, and are independent of the number of random numbers generated in other runs.

```
seed = 123456
```

reseed_day: if > -1, start each run with the same random seed and then reset the seed at day reseed_day. The effect is that the initial days will follow the same trajectory, but the simulations will follow independent trajectories starting on reseed_day. This permits estimation of conditional variance.

```
reseed_day = -1
```

office_size: maximum number of workers per office. If set to 0, then workplaces are not subdivided into offices.

```
office_size = 50
```

classroom_size: maximum number of students per classroom. If set to 0, then schools are not subdivided into classrooms.

```
school_classroom_size = 40
```

neighborhood: When deciding where to spend an agent's "neighborhood" time, there are parameters to control the probability of selecting a random cell within the "community", defined by the parameter `community_distance` (in km), and the probability that the agent goes to its "home" neighborhood (where the household is). The default parameters are:

```
# neighborhood activities
community_distance = 20
community_prob = 0.1
home_neighborhood_prob = 0.5
```

That is, 50% of the time, the neighborhood is the cell surrounding the household, and 10% of the time it is a random cell within 20km of home. The other 40% are distributed uniformly in the 8 cells immediately surrounding the home cell.

Disease Model Parameters

diseases: the number of diseases circulating in the population. Any number of diseases is allowed. Runtime and memory required is proportional to the number of diseases.

```
diseases = 1
```

Each disease is described by the following set of parameters, indexed by the disease number d, where d = 0,...,diseases-1.

primary_cases_file[d]: the file containing the number of primary cases to be injected into the simulation during each day.

```
primary_cases_file[0] = primary_case_schedule-0.txt
```

The **primary_cases_file[d]** follows the Multistrain Timestep Map input format.

trans[d]: the transmissibility of disease d relative to an arbitrary baseline.

```
trans[0] = 1.0
```

symp[d]: the probability of an infected person becoming symptomatic

```
symp[0] = 0.67
```

mortality_rate[d]: the probability of an infected person dying (Not currently implemented)

```
mortality_rate[0] = 0.00001
```

i**nfection_model[d] =** 0 or 1. Infection model 0 is a bifurcating model in which each infected agent passes through stages SEIR or SEiR, where "I" means infectious and symptomatic, and "i" means infectious but not symptomatic. Infection model 1 is a sequential model in which infected agents pass through the stages SEiIR. In any model, some stages may last for 0 days, except E, which always lasts at least 1 day.

```
infection_model[0] = 0
```

days_latent[d]: discrete cdf for number of days between becoming exposed and becoming infectious. With the values shown in the example below, there is an 80% chance of becoming infectious 1 day after exposure and a 20% chance of becoming infectious 2 days after exposure.

```
days_latent[0] = 3  0 0.8 1.0
```

days_asymp[d]: discrete cdf for number of days the agent is infectious but asymptomatic. With the values shown in the example below, the default setting , the agent may be asymptomatic between 3 to 6 days.

```
day_asymp[0] = 7   0.0  0.0  0.0  0.3  0.7  0.9  1.0
```

days_symp[d]: discrete cdf for number of days the agent is infectious and symptomatic. With the values shown in the example below, the default setting , the agent may be symptomatic between 3 to 6 days.

```
day_symp[0] = 7   0.0  0.0  0.0  0.3  0.7  0.9  1.0
```

immunity_loss_rate[d]: rate at which a person loses immunity after recovering from infection. If greater than 0.0, the number of days in state 'R' is drawn from an exponential distribution with parameter 1/immunity_loss_rate.

```
immunity_loss_rate[0] = 0
```

symp_infectivity[d]: multiplier for how infective a symptomatic agent is.

```
symp_infectivity[0] = 1.0
```

asymp_infectivity[d]: multiplier for how infective an asymptomatic agent is.

```
asymp_infectivity[0] = 0.5
```

residual_immunity_ages[d]:

```
residual_immunity_ages[0] = 0
```

residual_immunity_values[d]:

```
residual_immunity_values[0] = 0
```

pregnancy_prob_ages:

```
pregnancy_prob_ages = 0
```

pregnancy_prob_values:

```
pregnancy_prob_values = 0
```

at_risk_ages[d]:

```
at_risk_ages[d] = 0
```

at_risk_values[d]:

```
at_risk_values[d] = 0
```

prob_stay_home: the probability that a symptomatic agent stays home

```
prob_stay_home = 0.5
```

mutation_prob:

```
mutation_prob = 1 0.0
```

Contact Rates

The following parameters determine the number of potentially infective daily contacts between an infectious agent and a susceptible agent in a given type of location. The default values are calibrated for Allegheny County using the bifurcating infection model (infection_model = 0).

household_contacts[d]: contact rate for households.

```
household_contacts[d] = 0.19
```

neighborhood_contacts[d]: contact rate for neighborhoods.

```
neighborhood_contacts[0] = 42.32
```

school_contacts[d]: contact rate for schools.

```
school_contacts[0] = 15.83
```

workplace_contacts[d]: contact rate for workplaces.

```
workplace_contacts[0] = 1.66
```

By default, classroom contacts are double the school contacts, and office contacts are double the workplace contacts. These defaults are indicated as follows:

```
classroom_contacts[0] = -1
office_contacts[0] = -1
```

These defaults can be overridden if values other than -1 are provided in the params file.

weekend_contact_rate[d]: multiplier of neighborhood contacts on weekend. The default is to increase weekend contacts by 50%:

```
weekend_contact _rate[0] = 1.5
```

Transmission probabilities

The following parameters determine the probability that a potentially infective contact between an infectious agent and a symptomatic agent actually results in an infection. Transmission probabilities are defined for a given group in a given type of location. Each parameter is interpreted as a square matrix with the values given in row-order. The labels associated with the rows and columns (the groups) are specified in the comments, and are defined in the class associated with the parameter. For example, the definition of elementary students is defined in **School.cc**. The defaults are:

```
# groups = children adults
household_prob[d] = 4 0.6 0.3 0.3 0.4
neighborhood_prob[d] = 4 0.0048 0.0048 0.0048 0.0048

# groups = adult_workers
workplace_prob[d] = 1 0.0575
office_prob[d] = 1 0.0575
```

```
# groups = elem_students mid_students high_students teachers
school_prob[d] = 16 0.0435 0 0 0 0 0.0375 0 0 0 0 0.0315 0 0 0 0 0.0575
classroom_prob[d] = 16 0.0435 0 0 0 0 0.0375 0 0 0 0 0.0315 0 0 0 0
0.0575
```

Multistrain Timestep Map Format

The first line of the timestep map file specifies the format to be used. Currently `#line_format` is the only supported format. Future work may allow for some type of structured (key = value) format to permit more detailed specification of seeding behavior.

Any line beginning with `#` is interpreted as a comment and ignored. Every other line is interpreted as a seeding instruction and expected to follow the format:

```
start end attempts [ strain [ prob [ min [ lat lon radius ] ] ] ]
```

Mandatory Fields:

The first three fields (`start, end, attempts`) are mandatory. The others are optional, but, if present, must be given in the order above.

The `start` and `end` fields are indexed from zero and can be used to specify a range of days beginning on `start` and continuing to `end` (inclusive). To specify seeding on a single day, set `start` equal to `end`.

The `attempts` field determines the number of seeding attempts for the given range of time steps. If no further fields are present, this number of individuals are randomly chosen with replacement from the entire population and transmission of the disease is attempted. Note that sampling includes individuals who may already be infected; in this case the actual number of new seeds may be less than the number specified by `attempts`.

Optional Fields:

The `strain` field gives the numeric id of the strain to be seeded for this timestep. If the strain field is not given, seeds will be strain 0.

The `prob` field can be used to introduce some randomness into the number of seeding events attempted at the time step. With probability `1-prob` each of the attempts specified by `attempt` will be skipped.

The `min` field can be used to ensure that a minimum number of attempts actually result in transmission. If specified, individuals will continue (1000 additional times) to be selected from the population until `min` number of successful transmissions have been created. If 1000 additional selections from the population are insufficient to create the specified minimum number of transmissions, a warning is given and execution of the program continues.

The geographic area from which individuals are selected can be specified by giving the coordinates of a point (`lat`, `lon`) and a `radius` specified in kilometers. When enabled, random sampling is restricted to only those individuals whose households are located within the specified area.

Additional information on the timestep map format may be found in **README_Timestep_Maps**.

Intervention Parameters

School closure parameters

Parameters controlling to school closures:

```
# set to 1 if schools closed during summer
school_summer_schedule = 0

# summer schedule dates (format MM-DD)
school_summer_start = 06-01
school_summer_end = 08-31

school_closure_policy = none
# school_closure_policy = global
# school_closure_policy = individual

# number of days to keep a school closed
school_closure_period = 10

# delay after reaching any trigger before closing schools
school_closure_delay = 2

# day to close school under global policy
school_closure_day = 10
```

Vaccination control parameters

Parameters controlling to vaccination:

`vaccine_tracefile:` If the value is "none", no vaccine tracefiles are produced. Otherwise, a vaccine tracefile is produced for each run in the directory given the outdir parameter. A vaccine tracefile contains one record for each agent, giving the agent's vaccination history. Vaccine tracefiles are named vtrace1.txt, vtrace2.txt, etc.

```
vaccine_trace = none
```

`number_of_vaccine:` the number of types of vaccines that you would like to run in the simulation. There needs to be a set of vaccine parameters for each vaccine in the system or the simulation will end in error.

Default: 0

vaccine_prioritize_acip: Enable prioritization of vaccination by ACIP recommendations. This includes persons aged 0-24, people deemed at risk for complications for influenza (see at_risk_ages and at_risk_values keywords), pregnant women (see pregnancy_prob_ages and pregnancy_prob_values keywords), and people over age 64.

Default: 0

vaccine_prioritize_by_age: Enables prioritization of vaccination by age group. The age groups will be defined by the two following keywords.

Default: 0

vaccine_priority_age_low: If vaccine_prioritize_by_age is specified as 1, this specifies the lower limit of the prioritized age group inclusively.

Default 0

vaccine_priority_age_high: If vaccine_prioritize_by_age is specified as 1, this specifies the upper limit of the prioritized age group inclusively.

Default 100

vaccine_dose_priority: If there are multi-dose vaccines, this parameter defines prioritization of people getting multiple doses vs. people getting their first dose.

Possible values:

- 0 No Priority, first come first serve
- 1 Place people getting subsequent dose at the beginning of the queue
- 2 Mix in people getting subsequent dose with other priority vaccinations randomly
- 3 Place people getting subsequent dose at the end of the queue

Default

0

vaccine_capacity_file: This parameter specifies a file that defines how many agents the system has the capacity to vaccinate on a given day throughout the simulation. This may be more or less than the amount of vaccine available through production. This parameter is meant to allow the user to attenuate the system's ability to actually vaccinate people due to limitations in personnel, time and resources.

The format of this file follows a reduced Multistrain TimeStep file, with a format as follows:

```
Day_start       Capacity1
Day_change1     Capacity2
Day_change2     Capacity3
….
```

For example: If one wanted to define that for the first 3 days of the simulation, the system could vaccinate no one, then on days 4-10, it could vaccinate 10000 people per day, then dropping down to 5000 per day on day 11 through the rest of the simulation, the vaccine_capacity_file would look like this:

```
0   0
4   10000
11  5000
```

The next set of parameters need to be defined for every vaccine in the simulation, and they will all be indexed by the vaccine number they define (signified by X).

vaccine_number_of_doses[X]: Specifies the number of doses needed for vaccine X. There needs to be a dose specification for each dose indicated, or the simulation will end in error.

Default

1

vaccine_total_avail[X]: Specifies the total amount of doses of vaccine X available for the entire simulation.

Default

1000000000

vaccine_additional_per_day[X]: The amount of vaccine X produced each day and made available to the system. The amount of vaccine produced cannot exceed vaccine_total_avail[X], for the entire simulation.

Default

1000000

vaccine_starting_day[X]: The day to start producing vaccine X at the rate defined by vaccine_additional_per_day[X].

Default
0

The next set of parameters must be specified for each dose (specified by Y) of vaccine X.

vaccine_next_dosage_day[X][Y]: Specifies the day of the dosage schedule that the next dose should be taken. For instance, if the dose Z of a vaccine is to be taken 7 days after dose Y, then this parameter for dose Y would be seven. The last dose of a vaccine is always 0.

Default:
0

vaccine_dose_efficacy_ages[X][Y]
vaccine_dose_efficacy_values[X][Y]: These parameters specify the age map for defining the efficacy of vaccine X, dose Y. The values should be probabilities between 0 and 1 that specify the probability that a person of a certain age will become immune after taking this dose of vaccine.

Default:
vaccine_dose_efficacy_ages[0][0] = 2 0 100
vaccine_dose_efficacy_values[0][0] = 1 0.70

vaccine_dose_efficacy_delay_ages[X][Y]
vaccine_dose_efficacy_delay_values[X][Y]: These parameters specify the age map for defining the delay to efficacy of vaccine X, dose Y. The values should be integer numbers of days by age.

Default:
vaccine_dose_efficacy_delay_ages[0][0] = 2 0 100
vaccine_dose_efficacy_delay_values[0][0] = 1 14

Anti-virals parameters

number_antivirals = 0

Overnight Travel Parameters

Parameters controlling long-distance overnight travel:

```
# enable overnight travel (optional)
enable_travel = 0

# cdf of trip duration in days
travel_duration = 6 0 0.2 0.4 0.6 0.8 1.0
```

That is, the default probability for the duration of travel being *i* days is 0.2, for *i* = 1 to 5 days.

```
# distance threshold for overnight trips (in km)
min_travel_distance = 100.0

# trips per day assuming entire US population
max_trips_per_day = 1000000

# file containing list of sample trips
tripfile = trips.txt
```

The format of the tripfile is:
`COL1 ROW1 COL2 ROW2`
where `(COL1, ROW1)` give the global cell coordinates for one endpoint, and `(COL2, ROW2)` give the global cell coordinates for the other endpoint. The order of the endpoints in irrelevant.

```
# file with population estimate for each large cell
cell_popfile = cell_pop.txt
```

The format of the cell_popfile is:
`COL1 ROW1 POP`
where `(COL1, ROW1)` give the global cell coordinates for one cell and `POP` is the number of agents in that cell according to the overall U.S. synthetic population file.

Behavioral Parameters

For each health-related behavior, FRED requires several parameters to describe how the behavior is modeled in the population. The current set of behavior includes:

- stay_home_when_sick
- keep_child_home_when_sick
- accept_vaccine
- accept_vaccine_dose
- accept_vaccine_for_child
- accept_vaccine_dose_for_child

In the following, replace <behavior_name> with the name of the specific behavior.

```
# enable the behavior
<behavior_name>_enabled = 1

#### BEHAVIOR MARKET SEGMENTS
#
# BEHAVIOR STRATEGY 0 = ALWAYS REFUSE
# BEHAVIOR STRATEGY 1 = ALWAYS ACCEPT
# BEHAVIOR STRATEGY 2 = FLIP WEIGHTED COIN FOR EACH DECISION
# BEHAVIOR STRATEGY 3 = IMITATE PREVALENCE
# BEHAVIOR STRATEGY 4 = IMITATE CONSENSUS
# BEHAVIOR STRATEGY 5 = IMITATE COUNT
# BEHAVIOR STRATEGY 6 = HEALTH BELIEF MODEL
#
# Each distribution should add up to 100
<behavior_name>_strategy_distribution = 7 50 50 0 0 0 0 0

##### FLIP/IMITATE INITIAL PROBS AND DECISION FREQUENCY
<behavior_name>_min_prob = 0
<behavior_name>_max_prob = 1
<behavior_name>_frequency = 1

#### IMITATION THRESHOLDS
<behavior_name>_imitate_consensus_threshold = 0
<behavior_name>_imitate_count_threshold = 0

##### WEIGHTS FOR IMITATION
```

```
## ORDER IS HOUSEHOLD NEIGHBORHOOD SCHOOL WORK ALL
## Weights can be any real number.
##
<behavior_name>_imitate_prevalence_weights = 5 0 0 0 0 1
<behavior_name>_imitate_consensus_weights = 5 0 0 0 0 1
<behavior_name>_imitate_count_weights = 5 0 0 0 0 1

#### IMITATE UPDATE RATES: HOW RAIDLY TO CONFORM TO CONSULT OTHERS
<behavior_name>_imitate_prevalence_update_rate = 0.1
<behavior_name>_imitate_consensus_update_rate = 0.1
<behavior_name>_imitate_count_update_rate = 0.1
<behavior_name>_susceptibility_threshold = 2 0 0


#### HEALTH BELIEF MODEL PARAMETERS (EXPERIMENTAL)
<behavior_name>_severity_threshold = 2 0 0
<behavior_name>_benefits_threshold = 2 0 0
<behavior_name>_barriers_threshold = 2 0 0
<behavior_name>_memory_decay = 2 0 0
<behavior_name>_base_odds_ratio = 1
<behavior_name>_susceptibility_odds_ratio = 1
<behavior_name>_severity_odds_ratio = 1
<behavior_name>_benefits_odds_ratio = 1
<behavior_name>_barriers_odds_ratio = 1
```

Running FRED

The FRED program takes an optional command line argument, the name of the run-time parameters file:

```
% FRED parameter_file_name
```

If the argument is omitted the name "params" is assumed.

In addition, a set of scripts is provided for managing the process of running a large number of simulations with FRED.

Simulation Information Management System

There are several options for running FRED. The FRED executable is copied to the $FRED/bin directory after each make, so you can run FRED as follows from any working directory, assuming that you have added `$FRED_HOME/bin` to your path:

```
% FRED [paramfile [run_number [directory]]]
```

The arguments are optional from right to left. If all three arguments are given, FRED uses the given paramfile, runs a single replication with number "run_number", and writes output files to the given directory. The output directory is relative to the current working directory.

If the third argument is omitted, the output directory is taken from the runtime parameter "`outdir`", with default value `OUT`.

If both the second and third arguments are missing, run_number defaults to 1.

If all arguments are missing, paramfile defaults to "params".

Examples:

```
# run FRED on file params and write output files to ./OUT:
% FRED

# run FRED on file params.foo and write output files to ./OUT:
% FRED params.foo

# run FRED on file params with run number = 2
% FRED params 2

# run FRED on file params.foo
# with run number = 2 writing output files to ./OUT.foo:
% FRED params.foo 2 OUT.foo
```

Using the `run_fred` script for multiple realizations

The **`run_fred`** script is provided to perform multiple realizations (runs) in a local directory. Each run uses a distinct seed for the random number generator, so the results will vary from run to run. The format is:

```
% run_fred -p paramfile -d directory -s start_run -n end_run
```

The order of the arguments doesn't matter, and all arguments have default values:

```
-p "params"
-d ""
-s 1
-n 1
```

For example, the command:

```
% run_fred -p params -d FOO -s 1 -n 3
```

translates to a set of commands:

```
% FRED params 1 FOO > FOO/LOG1
% FRED params 2 FOO > FOO/LOG2
% FRED params 3 FOO > FOO/LOG3
```

after first creating directory FOO if necessary. The run_fred script also copies the params file into the output directory, for future reference.

If -d is not specified on the command line, FRED writes output files to the output directory specified in the **outdir** runtime parameters, which default to **OUT**. For example, if params does not specify an output directory, then

```
% run_fred -n 3
```

translates to:

```
% FRED params 1 OUT > OUT/LOG1
% FRED params 2 OUT > OUT/LOG2
% FRED params 3 OUT > OUT/LOG3
```

The random seed for each run is set based on the both the seed value in the params file and on the run number, so a collection of FRED runs can be executed in any order with the same results. For example, you should get the same results in the output directory from

```
% run_fred -n 20
```

as from:

```
% run_fred -n 10
% run_fred -s 11 -n 20
```

FRED runtime management scripts

The $FRED_HOME/bin directory includes several commands to manage the process of running FRED jobs. Commands exist for starting FRED jobs, reporting the status of those jobs, and organizing and reporting the results files. The bin directory contains the following commands:

`fred_job` -- runs FRED and stores all associated data in a results database
`fred_AR` -- report on the the attack rate of a simulation
`fred_clear_all_results` -- flush the results database
`fred_delete` -- delete a single job from the results database
`fred_display_plot` -- display one or more curves
`fred_jobs` -- show that status of all jobs in the results database
`fred_plot` -- plot a curve
`fred_plot_data` -- retrieve the data associated with a curve
`fred_report` -- create statistical summaries of output variables
`fred_status` -- report the status of a single job
`fred_sweep` -- run a set of simulation changing the value of a variable
`fred_tail` -- show the tail of the current output file
`get_distr` -- show the distribution of infection locations
`ch` -- change a parameter value in a params file
`p` -- print out the current params file

To use these commands, set the environmental variable `$FRED_HOME` to the location of your FRED distribution. Then add `$FRED_HOME/bin` to your path. The following are most likely to be the most useful commands when starting to use FRED.

Command descriptions:

`% fred_job [-p paramsfile | -k key | -c ]`
Run FRED with the given parameter file in a working directory created in the `$FRED_HOME/RESULTS` directory, and associate the working directory with the key. If the `-p` option is omitted, the file "`params`" is assumed. If the `-k` option is omitted, an internally created key is generated. In either case, a `<key,id>` pair is printed on standard output, where `<id>` is the identifier of directory associated with the run (i.e. `$FRED_HOME/RESULTS/JOB/<id>`).

`fred_job` will terminate if the user supplied key has already been used. If the -c (cache) arguments is specified, then if the params file duplicates a previous params file, fred_job associates the key with previous id, and does not re-run FRED. The script sets the STATUS of the request (see `fred_status` below). When FRED finishes, fred_job runs `stats` to collect data on the output variables in the outfile.

`% ch param_name value [ paramfile ]`
Edit the given paramfile (or "params" if no file is given) and add a line
`param_name = value`

First checks to see if the given `param_name` occurs in **params.default**.
Note: If the parameter contained a shell meta-character, you should enclose the parameter name in quotes.

`% fred_AR -k key`
Return mean and std dev of attack rate (AR) for run associated with key.

`% fred_clear_all_results`
Flush all the data from the results database.

`% fred_delete -k key`
Delete a single job from the results database. Example:

```
% fred_delete -k test1
KEY = test1  RUN = 15
You are about to delete /Users/gref/Desktop/FRED/RESULTS/RUN/15. This
cannot be undone.
Proceed? yes/no [no]
y
/Users/gref/Desktop/FRED/RESULTS/RUN/15 deleted
```

The -f flag forces deletion:

```
% fred_delete -f -k test1
/Users/gref/Desktop/FRED/RESULTS/RUN/15 deleted
```

% fred_display_plot -k key -v [S|E|I|R|s|C|c|M|A|r]
Run fred_plot and then opens the resulting plot file.

% fred_jobs
Show that status of all jobs in the results database. **For** example:

```
% fred_jobs
KEY = baseline           JOB =   1        STATUS = FINISHED Thu Sep 30
12:20:04 2010
KEY = baseline_trans[0]=0.9          JOB =   2        STATUS = FINISHED
Thu Sep 30 14:21:43 2010
KEY = baseline_trans[0]=1     JOB =   3          STATUS = FINISHED Thu
Sep 30 14:52:40 2010
KEY = baseline_trans[0]=1.1          JOB =   4        STATUS = RUNNING-
43 Thu Sep 30 15:07:35 2010
```

The dates shown for FINISHED jobs reflect the time that they finished.

% fred_plot -k key -v [S|E|I|R|s|C|c|M|A|r]
Create a plot of one or more of the indicated measures for the indicated run. The plot file (type PNG) is stored in RESULTS under the run's REPORT directory. Prints the full path to the plot file.

% fred_plot_data -k key -v [S|E|I|R|s|C|c|M|A|r]
Print the data for plotting the graph to standard output, in space delimited format:
`day mean stdev`

% fred_status -k key [-s secs]
Print the status of the FRED run associated with the given key. If `-s` option is given, repeats status report every `secs` seconds.

% fred_sweep key param lower_bound upper_bound increment
Run a set of simulation changing the value of a parameter. All 5 arguments are required. The arguments are:

key = a suffix for the parameter file that defines the scenario.

param = the name of the parameter you wish to sweep

lower_bound, upper_bound, increment are self-explanatory.

You must first create a file called params.<key> that sets up the rest of the parameters. For each value of the named parameter, the script creates a params file called params.<key>_<parameter>=<value> and executes the command:

% fred_job -p params.<key>_<parameter>=<value> -k <key>_<parameter>=<value>

Each fred_job command is executed in the foreground, so the jobs run one at a time.

Note: If the parameter contained a shell meta-character, you should enclose the parameter name in quotes. Example:

```
% fred_sweep baseline 'trans[0]' 0.9 1.1 0.1
```

has the effect of making three copies of the file `params.baseline`, changing the value of `trans[0]` in each one and executes

```
% fred_job -p params.baseline_trans[0]=0.9 -k baseline-trans[0]-0.9
% fred_job -p params.baseline_trans[0]=1 -k baseline-trans[0]-1
% fred_job -p params.baseline_trans[0]=1.1 -k baseline-trans[0]-1.1
```

% fred_tail -k key
Run the `tail -f` command on the current output file.

% get_distr
Show the distribution of infection locations.

% p
Print out the current params file.

% rt
Run regression test.

Notes for Developers

Contributed Code

FRED is intended to be a system that evolves over time to be the varied need of researchers in the infectious disease modeling field. We hope that developers will want to modify the code and add new features. If you do develop new features and want to share with the rest of the FRED community, please consider adding it to the official FRED distribution. We are happy to discuss this in more detail.

Coding Standards

The FRED team believes that coding standards in general make for cleaner, more readable code, and may help avoid certain pitfalls. We have tried to develop FRED according to the Google code standards:

http://google-styleguide.googlecode.com/svn/trunk/cppguide.xml

No claim is made that we have achieved complete success, but we have found the attempt helpful.

Regression Tests

FRED includes a number of regression tests that can be run after editing the code to help catch unintended changes. The $FRED_HOME/bin directory contains some scripts to support testing FRED:

`make_rt directory_name`	**-- make files for regression test**
`rt` [-p] [directory_name]	**-- run regression test**

The test directory tree is located at **$FRED_HOME/tests.** The tests for the FRED base code are located in subdirectory **base.** There are a few other test directories, and more will be added over time. Each test directory contains at least two files: **params.test** and **compare.** The **params.test** file contains the run-time parameters that test the given feature. The FRED script **rt** runs a few FRED simulations in the test directory, using the **params.test** file. The output is directed to subdirectory **OUT.TEST.** The **rt** script compares the output files in **OUT.TEST** with the files in subdirectory **OUT.RT.** The specific comparisons are up to the developer, and are found in the executable file **compare**, which is run by the **rt** script when the simulations are complete. If no errors are generated by **compare**, then FRED can be said to have passed this particular regression test.

The **rt** script takes two optional arguments:

```
% rt –p test_name
```

test_name should be the name of one of the directories in **$FRED_HOME/tests**. If this argument is omitted, test_name defaults to "base".

If the –p argument is given, **rt** will run the test simulations in parallel. If –p is given in must be the first argument.

The **rt** command can be run from any directory. It will temporarily change to the test directory to run FRED, and then return to the original directory.

To create a new regression test, do the following:

1. Create a new directory in $FRED_HOME/tests:

```
% mkdir $FRED_HOME/tests/foo
```

2. Create a params file in that directory.

3. Create an executable file called **compare** that implements whatever tests you wish to make on the resulting FRED output files in subdirectory OUT.TEST.

4. Run the script **make_rt** to create the target output file. These will be stored in subdirectory **OUT.RT.**

```
% make_rt foo
```

Test your regression test by running:

```
% rt foo
% rt –p foo
```

References

Beckman RJ., Baggerly K, McKay M. Creating synthetic baseline populations. Transportation Research Part A: Policy and Practice. 1996; 30(6): 415-429.

Cajka, JC, Cooley, PC, Wheaton, WD. Attribute Assignment to a Synthetic Population in Support of Agent-Based Disease Modeling RTI Press. 2010; http://www.rti.org/pubs/mr-0019-1009-cajka.pdf

Cooley P, Brown S, Cajka J, Chasteen B, Ganapathi L, Grefenstette J, Hollingsworth CR, Lee BY, Levine B, Wheaton WD, Wagener DK. The role of subway travel in an influenza epidemic: a New York City simulation. *J Urban Health*. 2011 Oct;88(5):982-95. PubMed PMID: 21826584; PubMed Central PMCID: PMC3191213

Wheaton, WD, Cajka, JC, Chasteen, BM, Wagener, DK, Cooley, PC, Ganapathi, L. Synthesized population databases: A U.S. geospatial database for agent-based models. RTI Press. 2009; http://www.rti.org/pubs/mr-0010-0905-wheaton.pdf

Appendix: FRED License Agreement

A license is hereby granted by University of Pittsburgh Graduate School of Public Health ("GSPH"), free of charge, to any person obtaining a copy of the software package called FRED and associated documentation files (the "Software"), to use, copy, modify and merge copies of the Software, subject to the following conditions:

1. You acknowledge and agree that the license granted hereunder is personal to you, and you will not under any circumstances sell, give, disclose, lend, or otherwise distribute the Software to third parties. You further agree that you will not use the Software for commercial purposes.

2. You acknowledge and agree that GSPH retains all ownership rights, including copyright rights in the Software and that by entering into this license, you do not acquire any such rights in the Software.

3. You acknowledge and agree that THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

4. If the Software is used to obtain a result, and that result is published in the public literature, then you agree to acknowledge its use of the Software in the following citation:

Copyright © 2011, University of Pittsburgh Graduate School of Public Health, John Grefenstette, Shawn Brown, Roni Rosenfield, Alona Fyshe, David Galloway, Nathan Stone, Bruce Lee, Phil Cooley, William Wheaton, Thomas Abraham, Jay DePasse, Anuroop Sriram, and Donald Burke.

5. You agree to indemnify and hold harmless GSPH from and against all damages, liabilities, attorney fees, and costs arising out of your use of the Software.