Stride project

User Manual

Version 1.0 (November 22, 2018) .

Centre for Health Economics Research & Modeling of
Infectious Diseases, Vaccine and Infectious Disease
Institute, University of Antwerp.

Modeling of Systems and Internet Communication,
Department of Mathematics and Computer Science,
University of Antwerp.

Interuniversity Institute for Biostatistics and statistical
Bioinformatics, Hasselt University.

Willem L, Kuylen E & Broeckhove J

Contents

1 Introduction 2

2 Software 4

2.1 System Requirements 4

2.2 Installation 4

2.3 Documentation 5

2.4 Directory layout 5

2.5 File formats 5

2.6 Testing 6

2.7 Results 6

2.8 Protocol Buffers 7

3 Simulator 9

3.1 Workspace 9

3.2 Run the simulator 11

3.3 Generating a population and geographical grid 11

3.4 Using the Calibrator 12

3.5 Python Wrapper 13

4 Code **14**

4.1 GenGeoPop 14

CHAPTER 1

Introduction

This manual provides a brief description of the Stride software and its features. Stride stands for **S**imulation of **tr**ansmission of **i**nfectious **d**is**e**ases and is an agent-based modeling system for close-contact disease transmission developed by researchers at the University of Antwerp and Hasselt University, Belgium. The simulator uses census-based synthetic populations that capture the demographic and geographic distributions, as well as detailed social networks. Stride is an open source software. The authors hope to make large-scale agent-based epidemic models more useful to the community. More info on the project and results obtained with the software can be found in: *"Willem L, Stijven S, Tijskens E, Beutels P, Hens N & Broeckhove J. (2015) Optimizing agent-based transmission models for infectious diseases, BMC Bioinformatics, 16:183"* [**?**].

The model population consists of households, schools, workplaces and communities, which represent a group of people we define as a "ContactPool". Social contacts can only happen within a ContactPool. When school or work is off, people stay at home and in their primary community and can have social contacts with the other members. During other days, people are present in their household, secondary community and a possible workplace or school.

We use a *Simulator* class to organize the activities from the people in the population. The ContactPools in a population are grouped into ContactCenters (e.g. the different classes of a school are grouped into one K12School ContactCenter). These ContactCenters are geographically grouped into a geographical grid (sometimes called GeoGrid)

The *ContactHandler* performs Bernoulli trials to decide whether a contact between an infectious and susceptible person leads to disease transmission. People transit through Susceptible-Exposed-Infected-Recovered states, similar to an influenza-like disease. Each *ContactPool* contains a link to its members and the *Population* stores all personal data, with *Person* objects. The implementation is based on the open source model from Grefenstette et al. [**?**]. The household, workplace and school clusters are handled separately from the community clusters, which are used to model general community contacts. The *Population* is a collection of *Person* objects.

CHAPTER 2

Software

2.1 System Requirements

Stride is written in C++ and portable over all platforms that have a sufficiently recent version of the GNU C++ compiler. The following tools needs to be installed:

- g++
- make
- CMake (>= v3.10)
- Boost (>= 1.66 is known to work)
- Python (optional, for automatization)
- Doxygen (optional, for documentation)
- LaTeX (optional, for documentation)

2.2 Installation

To install the project, first obtain the source code by cloning the repository to a directory . The build system for Stride uses the CMake tool. This is used to build and install the software at a high level of abstraction and almost platform independent (see http://www.cmake.org/). The project includes the conventional make targets to "build", "install", "test" and "clean" the project. There is one additional target "configure" to set up the CMake/make structure that will actually do all the work. For those users that do not have a working knowledge of CMake, a front end Makefile has been provided that invokes the appropriate CMake commands. More details on building the software can be found in "INSTALL.txt" in the source folder.

2.3 Documentation

The Application Programmer Interface (API) documentation is generated automatically using the Doxygen tool (see `www.doxygen.org`) from documentation instructions embedded in the code .

The user manual distributed with the source code has been written in LaTeX(see `www.latex-project.org`).

2.4 Directory layout

The project directory structure is very systematic. Everything used to build the software is stored in the root directory:

- `main`: Code related files (sources, third party libraries and headers, ...)
  - `main/<language>`: source code, per coding language: cpp (for C++), python, R,
  - `main/resources`: third party resources included in the project:
- `doc`: documentation files (API, manual, ...)
  - `doc/doxygen`: files needed to generate the reference documentation with Doxygen
  - `doc/latex`: files needed to generate the user manual with Latex
- `test`: test related files (scripts, regression files, ...)

2.5 File formats

The Stride software supports different file formats:

CSV
Comma separated values, used for population input data and simulator output.

HDF5
Hierarchical Data Format 5, designed to store and organize large amounts of data.

JSON
JavaScript Object Notation, an open standard format that uses human-readable text to transmit objects consisting of attribute-value pairs. (see `www.json.org`)

TXT
Text files, for the logger.

XML
Extensible Markup Language, a markup language (both human-readable and machine-readable) that defines a set of rules for encoding documents.

Proto
Protocol Buffers, used for exporting and importing the generated population and geographical grid.

2.6 Testing

Unit tests and install checks are added to Stride based on Google's "gtest" framework and CMake's "ctest" tool. In addition, the code base contains assertions to verify the simulator logic. They are activated when the application is built in debug mode and can be used to catch errors at run time.

2.7 Results

The software can generates different output files:

cases.csv
Cumulative number of cases per day.

summary.csv
Aggregated results on the number of cases, configuration details and timings.

person.csv
Individual details on infection characteristics.

logfile.txt
Details on transmission and/or social contacts events.

gengeopop.proto
Generated population and geographical grid.

calibration_data.json
Results of the calibration component.

2.8 Protocol Buffers

Protocol Buffers[1] is a library used for serializing structured data. We use it to read and write the population generated by GenGeoPop (see 4.1). It uses an interface description language that describes the structure of the data we want to store, in this case the GeoGrid. The file that describes how the GeoGrid is structured inside the generated binary file is located at: `main/cpp/gengeopop/io/proto/geogrid.proto` This file is then used to generate the C++ code necessary to read and write our GeoGrid in the structure we provided. We include this generated code in the project so it's not necessary to install the `protobuf-c` package[2] in order to compile Stride. These generated source files can be found in the same folder. If you want to change the structure, for example due to changes in the way the population is generated, you will need to install the previously mentioned package and generate the necessary code again. To make it easier to do this, we provided a CMake parameter and target that will generate the code and copy it to the source directory respectively. For this to work, Stride needs to be compiled using the "`STRIDE_GEN_PROTO=true`" argument. This will generate the code based on the `geogrid.proto` file and use that instead of the version included in the source directory. If you then want to copy this code to the correct location in the source, you can use "`make proto`". If the version of `protobuf-c` you're using is significantly newer than the `protobuf` included in this repository, you might also need to update the files stored `main/resources/lib/protobuf`. At the time of writing, this is done in the following way, although this may be subject to change:

- Copy all files from `https://github.com/protocolbuffers/protobuf/tree/master/src/google/protobuf` to `main/resources/lib/protobuf/google/protobuf`.
- Copy the `libprotobuf_files` and `libprotobuf_includes` from `https://github.com/protocolbuffers/protobuf/blob/master/cmake/libprotobuf.cmake` and `https://github.com/protocolbuffers/protobuf/blob/master/cmake/libprotobuf-lite.cmake`
- Remove the prefix `$protobuf_source_dir/src/` from these listings.
For example `$protobuf_source_dir/src/google/protobuf/any.cc` becomes `google/protobuf/any.cc`.
- Edit the `CMakeLists.text` at `main/resources/lib/protobuf` and replace the current values of `libprotobuf_files` and `libprotobuf_includes` with the ones we just obtained.

[1] https://developers.google.com/protocol-buffers/
[2] https://github.com/protobuf-c/protobuf-c

The classes that are responsible for reading and writing a GeoGrid to an `istream` containing the serialized data are the `GeoGridProtoReader` and `GeoGridProtoWriter` respectively.

CHAPTER 3

Simulator

3.1 Workspace

By default, Stride is installed in `./target/installed/` inside de project directory though this can be modified using the CMakeLocalConfig.txt file (example is given in `./src/main/resources/make`). Compilation and installation of the software will create the following files and directories:

- Binaries in directory `<project_dir>/bin`
  - *stride*: executable.
  - *gtester*: regression tests for the sequential code.
  - *gengeopop*: generates the population and geographical grid.
  - *calibration*: tool to run the simulator multiple times to generate statistical data. This can be used to calibrate test outcomes.
  - *wrapper_sim.py*: Python simulation wrapper
- Configuration files (xml and json) in directory `<project_dir>/config`
  - *run_default.xml*: default configuration file for Stride to perform a Nassau simulation.
  - *run_generate_default.xml*: default configuration file for Stride to first generate a population and geographical grid and then perform a Nassau Simulation.
  - *run_import_default.xml*: default configuration file for Stride to first import a population and geographical grid and then perform a Nassau Simulation.

  - *run_regions_default.xml*: default configuration file for Stride to first generate several regions with their own population and geographical grid and then perform a Nassau Simulation.
  - *run_regions_import.xml*: default configuration file for Stride to first import several regions with their own population and geographical grid and then perform a Nassau Simulation.
  - *run_miami_weekend.xml*: configuration file for Stride to perform Miami simulations with uniform social contact rates in the community clusters.
  - *wrapper_miami.json*: default configuration file for the wrapper_sim binary to perform Miami simulations with different attack rates.
  - ...
- Data files (csv) in directory `<project_dir>/data`
  - *belgium_commuting*: Belgian commuting data for the active populations. The fraction of residents from "city_depart" that are employed in "city_arrival". Details are provided for all cities and for 13 major cities.
  - *belgium_population*: Relative Belgian population per city. Details are provided for all cities and for 13 major cities.
  - *flanders_cities*: Cities and municipalities in Flanders with coordinates and population figures based on samples. These relative population figures are used for assigning residencies and domiciles based on a discrete probability distribution.
  - *flanders_commuting*: Relative commuting information between cities and communities. Since this data is relative, the total number of commuters is a derived parameter, based on the fraction of the total population that is commuting.
  - *contact_matrix_average*: Social contact rates, given the cluster type. Community clusters have average (week/weekend) rates.
  - *contact_matrix_week*: Social contact rates, given the cluster type. Community clusters have week rates.
  - *contact_matrix_week*: Social contact rates, given the cluster type. Primary Community cluster has weekend rates, Secondary Community has week rates.
  - *disease_xxx*: Disease characteristics (incubation and infectious period) for xxx.
  - *holidays_xxx*: Holiday characteristics for xxx.
  - *pop_xxx*: Synthetic population data extracted from the 2010 U.S. Synthetic Population Database (Version 1) from RTI International for xxx [? ?].
  - *ref_*2011: Reference data from EUROSTAT on the Belgian population of 2011. Population ages and household sizes.
  - *ref_fl*2010*_xxx*: Reference data on social contacts for Belgium, 2011.
- Documentation files in directory `./target/installed/doc`
  - Reference manual
  - User manual

3.2 Run the simulator

From the workspace directory, the simulator can be started with default configuration using the command "`./bin/stride`". Settings can be passed to the simulator using one or more command line arguments:

- `-c` or `--config`: The configuration file.

3.3 Generating a population and geographical grid

From the workspace directory, the generation of a population and geographical grid (sometimes called GeoGrid) can be started with the default configuration using the command "`./bin/gengeopop`". The following configuration options are available:

`--populationSize`
The size of the population to generate. By default a population of `6000000` is generated.

`--fracActive`
The fraction of people who are active, i.e. who are employed or students. By default `0.75` is used.

`--fracStudentCommuting`
The fraction of students commuting. By default `0.5` is used.

`--fracActiveCommuting`
The fraction of active people who commutes. By default `0.5` is used.

`--frac1826Students`
The fraction of 1826 years which are students. By default `0.5` is used.

`--household`
The file to read the household profiles from.

`--commuting`
The file to read the commuting information from.

`--cities`
The file to read the cities from.

`--output`
The file to write the GeoGrid to. By default this is `gengeopop.proto`

`--state`
The state to be used for initializing the random engine. This can be used to continue with the same state when generating multiple regions.

`--seed`
: The seed to be used for the random engine. The default is `"1,2,3,4"`.

`--loglevel`
: The loglevel to use, by default this is `info`.

3.4 Using the Calibrator

The Calibrator is a tool designed to calibrate the scenario tests. It can also be used for running a simulation multiple times to gather statistical data. This data can then be written to a file or be used for generating boxplots. The following configuration options are available:

`--config`
: Specifies the run configuration parameters to be used for the simulation. If this is provided multiple times, the calibration is performed on all given simulations. It may be either `-c file=<file>` or `-c name=<name>`. The first option can be shortened to `-c <file>`, the second option accepts `TestsInfluenza`, `TestsMeasles` or `BenchMeasles` as `<name`.

`--testcases`
: Instead of providing the configuration files, you can also select multiple testcases to use for the simulation runs. The default is `influenza_a, influenza_b, influenza_c, measles_16` and `r0_12`.

`--multiple`
: The amount of simulations to run for each testcase. For each simulation, a different seed will be used.

`--single`
: Run the simulations with the fixed seeds to determine the exact values.

`--output`
: Write the results of the calibration to a file with given filename. This resulting file contains for each configuration and each step in the simulation the mean, standard deviation, exact value using the default seed and the values found with other seeds. These values depend on the selected options, specifically `--multiple` and `--single`.

`--write`
: Write boxplots to files in the current directory. This creates an image for each config or testcase.

`--display`
: Display the boxplots for the last step.

`--displayStep`
: Display the boxplots for a specified step.

Examples:
To find the exact values for the testcases and write these to a file:

```
calibration -s -o out.json
```

To run a configuration file 10 times with a random seed and display the generated boxplot for the last step in the simulation:

```
calibration -c run_default.xml -m 10 -d
```

To run the testcase `influenza_a` 10 times, write the results to a file and for each step in the simulation write a boxplot to a file:

```
calibration -t influenza_a -m 10 -w -o out.json
```

3.5 Python Wrapper

A Python wrapper is provided to perform multiple runs with the C++ executable. The wrapper is designed to be used with .json configuration files and examples are provided with the source code. For example:

```
./bin/wrapper_sim --config ./config/wrapper_default.json
```

will start the simulator with each configuration in the file. It is important to note the input notation: values given inside brackets can be extended (e.g., “rng_seeds”=[1,2,3]) but single values can only be replaced by one other value (e.g., “days”: 100).

CHAPTER 4

Code

4.1 GenGeoPop

4.1.1 Background

To explain the algorithms used for generating the geography of the countries and their respective population, we have to introduce some background concepts:

ContactPool : A pool of persons that potentially come in contact with each other. If they actually do and if a transmission of an infection occurs depends on the algorithms in the simulator. We distinguish different types of ContactPools associated with work, school, community and households. Note that there is a difference between the size of a school (for example 500 students) and the size of the ContactPools inside the school (20 students).

ContactCenter : A group of one or more ContactPools of the same type. This allows for a single College to contain multiple classes for example. It is also used to contain a single ContactPool, for example in the case of a Household.

K-12 student : Part of the population from 3 until 18 years of age that are obligated (at least in Belgium) to attend a school. Students that skip or repeat years are not accounted for.

College student : Persons older than 18 and younger than 26 years of age that attend a higher education. For simplicity we will group all forms of higher eduction into the same type of ContactCenter, a College. A fraction of college students will attend a college "close to home" and the others will attend a college “far from home”. Most higher educations don’t last for 8 years, but using this number we compensate for changes in the field of study, doctoral studies, advanced masters and repeating a failed year of study.

Employable : We consider people of ages 18 to 65 to be employable. A fraction of people between 18 and 26 will attend a college, and the complementary fraction will be employable.

Active population : The fraction of the employable part of the population that is actually working. A fraction of these workers will work “close to home” and the complementary fraction will commute to a workplace further away.

Household profile : The composition of households in terms of the number of members and their age is an important factor in the simulation. In this case the profile is not defined by the age of its members or fractions, but through a set of reference households. This set contains a sample of households which are representative of the whole population in their composition.

GeoGrid locations : Our data only allows for a very limited geographical resolution. We have the longitude and latitude of cities and municipalities (a distinction we will not make), which we shall use to create GeoGrid locations for the domicile of the households. All households in the same location are mapped to the coordinates of the location’s center. These locations with corresponding coordinates will form a grid that will contain the complete simulation area.

4.1.2 Generating the geography

We start by generating the geographical component, a GeoGrid. This contains locations with an id, name, province, coordinates and nominative population figures. These locations contain multiple ContactCenters, like Schools and Households, which in turn contain ContactPools. This structure is internally generated by several “Generators” and will afterwards be used by “Populators” to fill the ContactPools. The different types of ContactCenters are created by a different partial generator for each type and added separately to the GeoGrid. We construct the following types of ContactCenters:

Households : The number of households is determined by the average size of a household in the reference profile and the total population size. These generated households are then assigned to a location by a draw from a discrete distribution based on the relative population size in each location.

K-12 Schools : These schools contain on average 500 students, with an average of 20 students per class, which corresponds to 25 ContactPools. The total amount of schools in the region is determined by the size of the population and the fraction of those people who attend a K-12 school. The assignment of the schools to the locations is analogous to that of households.

Colleges : These contain an average of 3000 students and 150 students per ContactPool. They are exclusively assigned to the 10 biggest locations in terms of population size. Within those 10 locations we again use a discrete distribution based on the relative population of the city compared to the total population in these top 10 cities.

Workplaces : The assignment of workplaces is analogous to that of households, but now we account for commuting information to determine the actual number of workers in a location. We find this amount by the working people who live there plus the people that commute to this location minus the ones that live there but commute away. A workplace contains on average 20 people.

Communities : We create both primary and secondary communities, each containing 2000 persons from all ages and occupations. The assignment is again analogous to that of households.

4.1.3 Generating the population

After creating the structures that will allow people to come in contact with each other and spread possible infections, we have create the population itself and determine the different ContactPools they will be in. The persons are created based on the Household profiles in the HouseholdPopulator. Analogous to the creation of the ContactCenters, we have a partial populator that will populate its own type of ContactPool:

Households : To create the actual persons, we draw a random household from the list of sample households and use that as a template for the number of members and their ages. We simply do this for each household in the GeoGrid, since we already determined the locations and amount of households while generating the geography.

K-12 Schools : We start by finding all schools within a 10km radius of the household location. If this list is empty, we double this searching radius until it's no longer empty. We then choose a random ContactPool from the combined list of ContactPools in the found schools, even if this ContactPool now has more students than average (20).

Colleges : The students who study "close to home" are assigned to a college in a way analogous to the assignment to K-12 schools. For the ones that study further from home we first determine the list of locations where people from this locations commute to and choose one of these locations by use of a discrete distribution based on the relative commuting information. After we have chosen a location, we randomly choose a ContactPool at a college in this location and assign it to the commuting student.

Workplaces : We first decide if the person actually has a job based on the unemployment parameter. We assign a workplace to an active worker that works "close to home" in an analogous way as the assignment of K-12 schools to students. For the commuting workers we use a technique analogous to that of the commuting college students.

Communities : The communities we can choose from at a location is determined in an analogous way to the K-12 schools. In primary communities we simply choose a random ContactPool from the list of Communities for each person in a household. In secondary communities however, we assign complete households to the ContactPools instead of each person in the household separately to a ContactPool.