Stride project

User Manual

Version 1.0 (June 19, 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.

Updated by Niels Aerens, Thomas Avé, Tobia De Koninck and Robin Jadoul as part of their Bachelor dissertation.

Willem L, Kuylen E & Broeckhove J

Contents

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 **dis**e**a**ses 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 "cluster". Social contacts can only happen within a cluster. 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, secundary community and a possible workplace or school.

We use a *Simulator* class to organize the activities from the people in an *Area*. The Area class has a *Population*, different *ContactPool* objects and a *ContactHandler*. 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 the GNU C++ compiler. To use the visualization tools Qt is needed (at least version 5.9), besides that the following tools needs to be installed:

- g++
- make
- CMake (>= v3.10)
- Boost
- 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 or download a zip file with all project material from the GitHub website and de-compress the archive. 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 directory `./src`:

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

map.png
Image of the map shown in the MapViewer component.

calibration_data.json
Results of the calibration component.

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.
  - *guilauncher*: tool to edit a simulation configuration and execute it
  - *mapviewer*: tool to visualize the population and geographical grid, can be directly used from *stride* and *guilauncher*.
  - *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.
  - *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_fl2010_xxx*: Reference data on social contacts for Belgium, 2011.
  - *DE_cities.csv*: German cities, fetched from [1].
  - *FR_cities.csv*: French cities, fetched from [2].
  - *NL_cities.csv*: Dutch cities, fetched from [3].
  - *wallonia_cities.csv*: Wallonia cities, fetched from [4].

1
2
3
4

  - *submunicipalities.csv*: Belgian sub-municipalities, fetched from [5].

- 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.
- `-r` or `--r0`: To obtain the basic reproduction number, no tertiary infections.

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.

[5]

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

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

`--subMinicipalities`
: The file to read the sub-municipalities 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.

`--rng_type`
: The type of random engine to use by default `mrg2` is used. Available options are: `mrg2`, `mrg3`, `yarn2`, `yarn3`, `lgc64` and `lgc64_shift`.

`--seed`
: The seed to be used for the random engine. The default is `0`.

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

3.4 Using the MapViewer

The MapViewer component can be used to explore a generated GeoGrid. From the workspace directory, the MapViewer can be started by using "`./bin/mapviewer`". No additional command line options are available. To open a GeoGrid use the menu option `File -> Open`. After loading the file, the viewport can be changed so that all markers are visible (`View -> Fit viewport`). The circular markers indicate a city or sub-municipality. The square markers indicate the "parent" city of sub-municipalities, they don't have a population. By clicking on the square marker, all the sub-municipalities will be selected. When one ore more cities is selected the details are shown in the right sidebar. For example a list of ContacCenters in the current selected Location. When a ContactCenter is clicked, the ContactPools of the clicked ContactCenter will be shown. When holding the Ctrl key and clicking multiple markers all clicked markers will be selected. Multiple cities in a rectangle can be selected by holding the control key and left mouse button while dragging. Clicking on the map will deselect all cities. Hold the Alt key while panning the map to prevent deselection. Ctrl+ A can be used to select all cities, while Ctrl + F can be used to fit the viewport of the map so all cities are visible. To show commutes of a city, first enable the `View -> Show Commutes` option. The commutes between the selected cities will be shown. While simulating a blue marker indicates that no one is infected at that location. As soon as some people get infected the marker will become green, to eventually become red when more people are infected. The visible area of a map can be exported to an image file using `File -> Export to image`.

When using the main Stride executable the mapviewer can be opened by passing the `-v` option.

3.5 Using the GuiLauncher

With the GuiLauncher component one can open a configuration file, change the parameters and then run the simulator. It's also possible to select which viewers to use (e.g. the MapViewer) and whether to use the GuiContrller (3.6). The changed configuration file can also be saved.

From the workspace directory, the GuiLauncher can be started by using "`./bin/mapviewer`". No additional command line options are available. The GuiLauncher should be straightforward to use.

3.6 Using the GuiController

The GuiController can be started when using the GuiLauncher. It makes it possible to control the simulation run. After clicking the "Start" button the simulation will start. The time between each simulated day can be changed by editing the value of the input field. The default is two seconds. It is also possible to advance the simulation multiple days, by choosing the amount of days and clicking the "Multi-Step" button. The "Step" button will always simulate one day.

3.7 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
```

3.8 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 quality

To ensure good code quality all compiler warnings were fixed our in some rare cases suppressed. The following compilers should give no warnings:

- GCC 6.4.0 on Linux
- GCC 7.3.0 on Linux
- GCC 7.3.1 on Linux
- GCC 8.1.1 on Linux
- Clang 5.0.0 on Linux
- Clang 5.0.1 on Linux
- Clang 5.0.2 on Linux
- Clang 6.0.0 on Linux
- AppleClang 8.1.0 on Darwin-16.7.0
- AppleClang 9.0.0 on Darwin-16.7.0
- AppleClang 9.1.0 on Darwin-17.4.0

Clang-format was used to ensure a consistent code-style throughout the project. The used clang-format is 5.0.1.

The GUI's were manually tested on a macOS High Sierra system. We noticed that on one system, out of our control, an extra titlebar appeared. We were unable to reproduce this. The used Qt version is