Indismo 3 project

User Manual

Version 3.0 (September 23, 2015) .

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, Stijven S & Broeckhove J

Contents

CHAPTER 1

Introduction

This manual provides a brief description of the indismo software and its features. Indismo is an open source 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. Indismo is open source in the hope of making 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 districts, which represent a group of people we define as a "cluster". Social contacts can only happen within a cluster. At night, people are present in their household and home district and can make social contacts with the other members. During day-time, people can stay at home or be assigned to a workplace or school in a specific district.

We use a *Simulator* class to organize the activities from the people in an *Area*. The Area class has a *Population*, different *Cluster* 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 *Cluster* 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. [2]. The household, workplace and school clusters are handled separately from the district clusters, which are only used to model general community contacts. The *Population* is a collection of *Person* objects.

CHAPTER 2

Software

2.1 System Requirements

Indismo is written in C++ and portable over all platforms that have the GNU C++ compiler. De software has no dependencies on external libraries. The following tools needs to be installed:

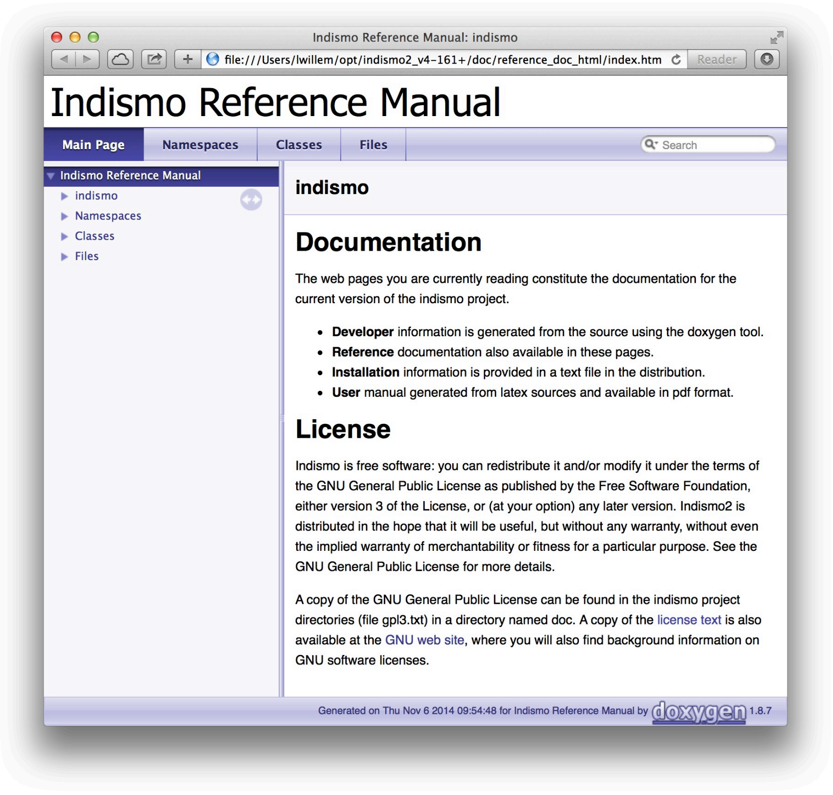
- g++
- make
- CMake
- Python (optional, for automatization)
- Doxygen (optional, for documentation)
- LaTeX (optional, for documentation)

2.2 Installation

To install the project, first obtain the source code by cloning the repository to a directory (e.g., “git clone https://bitbucket.org/indismo/indismo”) or download a zip file with all project material from the Bitbucket website and de-compress the archive. The build system for indismo 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 Doxygen from documentation instructions embedded in the code (see www.doxygen.org). The developer documentation is written in Doxygen syntax and can be generated in HTML format. Figure 2.1 presents the home page of the API documentation. The user manual distributed with the source code is written in LaTeX(see www.latex-project.org).


Indismo Reference Manual: indismo
file:///Users/lwillem/opt/indismo2_v4-161+/doc/reference_doc_html/index.htm
Reader
Main Page
Namespaces
Classes
Files
Search
Indismo Reference Manual
indismo
Namespaces
Classes
Files

Indismo Reference Manual

indismo

Documentation

The web pages you are currently reading constitute the documentation for the current version of the indismo project.

- **Developer** information is generated from the source using the doxygen tool.
- **Reference** documentation also available in these pages.
- **Installation** information is provided in a text file in the distribution.
- **User** manual generated from latex sources and available in pdf format.

License

Indismo is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version. Indismo2 is distributed in the hope that it will be useful, but without any warranty, without even the implied warranty of merchantability or fitness for a particular purpose. See the GNU General Public License for more details.

A copy of the GNU General Public License can be found in the indismo project directories (file gpl3.txt) in a directory named doc. A copy of the license text is also available at the GNU web site, where you will also find background information on GNU software licenses.

Generated on Thu Nov 6 2014 09:54:48 for Indismo Reference Manual by doxygen 1.8.7

Figure 2.1: Screenshot of API documentation generated with Doxygen.

2.4 Directory layout

The project directory structure is designed following maven conventions. Everything used to build the software is stored in directory `./src`:

- `src/main`: Code related files (sources, third party libraries and headers, ...)
  - `src/main/"language"`: source code, per coding language
  - `src/main/resources`: third party resources
- `src/doc`: documentation files (API, manual, ...)
  - `src/doc/"tool"`: files per document processing tools
- `src/test`: test related files (scripts, regression files, ...)

Every artefact during the build procedure is generated in directory `./target` and is completely removed when the project is cleaned.

2.5 File formats

The indismo software supports two file formats:

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

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`)

2.6 Testing

Unit tests and install checks are added to indismo 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 generates two output files:

Log
Cumulative number of cases per day.

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

CHAPTER 3

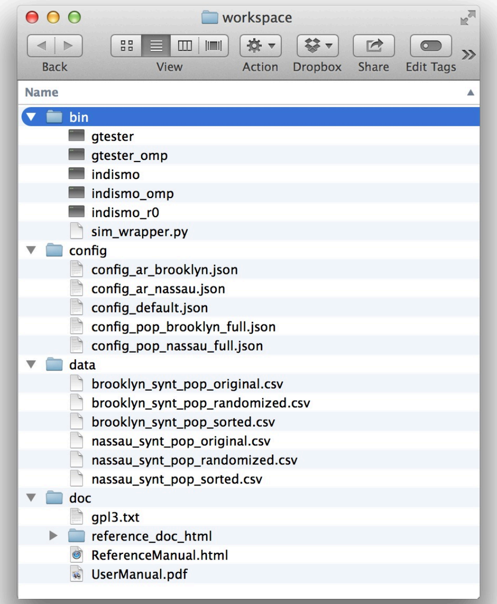
Simulator

3.1 Workspace

By default, indismo 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: (illustrated in Figure 3.1):

- Binaries in directory `./target/installed/bin`
  - *indismo*: sequential executable.
  - *indismo_omp*: OpenMP executable.
  - *indismo_r0*: executable to measure the basic reproduction number (R_0).
  - *gtester*: regression tests for the sequential code.
  - *gtester_omp*: regression tests for the OpenMP code.
  - *sim_wrapper.py*: Python simulation wrapper
- Configuration files (json) in directory `./target/installed/config`
  - *config_ar_alaska.json*: configuration file for the sim_wrapper to perform Alaska simulations with different attack rates.
  - *config_ar_brooklyn.json*: configuration file for the sim_wrapper to perform Brooklyn simulations with different attack rates.
  - *config_ar_nassau.json*: configuration file for the sim_wrapper to perform Nassau simulations with different attack rates.

- Input data files (csv) in directory `./target/installed/data`
  - *alaska_synt_pop_sorted*: Synthetic population data extracted from the 2005-2009 U.S. Synthetic Population Database (Version 1) from RTI International for Alaska. The person data is sorted according to day cluster (first) and household (second).
  - *brooklyn_synt_pop_sorted*: Synthetic population data extracted from the 2010 U.S. Synthetic Population Database (Version 1) from RTI International for Brooklyn, New York [3, 4]. The person data is sorted according to day cluster (first) and household (second).
  - *nassau_synt_pop_sorted*: Synthetic population data extracted from the 2010 U.S. Synthetic Population Database (Version 1) from RTI International for Nassau, New York [3, 4]. The person data is sorted according to day cluster (first) and household (second).
- Documentation files in directory `./target/installed/doc`
  - Reference manual
  - User manual



```
▼ bin
    gtester
    gtester_omp
    indismo
    indismo_omp
    indismo_r0
    sim_wrapper.py
▼ config
    config_ar_brooklyn.json
    config_ar_nassau.json
    config_default.json
    config_pop_brooklyn_full.json
    config_pop_nassau_full.json
▼ data
    brooklyn_synt_pop_original.csv
    brooklyn_synt_pop_randomized.csv
    brooklyn_synt_pop_sorted.csv
    nassau_synt_pop_original.csv
    nassau_synt_pop_randomized.csv
    nassau_synt_pop_sorted.csv
▼ doc
    gpl3.txt
  ▶ reference_doc_html
    ReferenceManual.html
    UserManual.pdf
```

Figure 3.1: Screen shot of the workspace directory.

3.2 Run the simulator

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

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

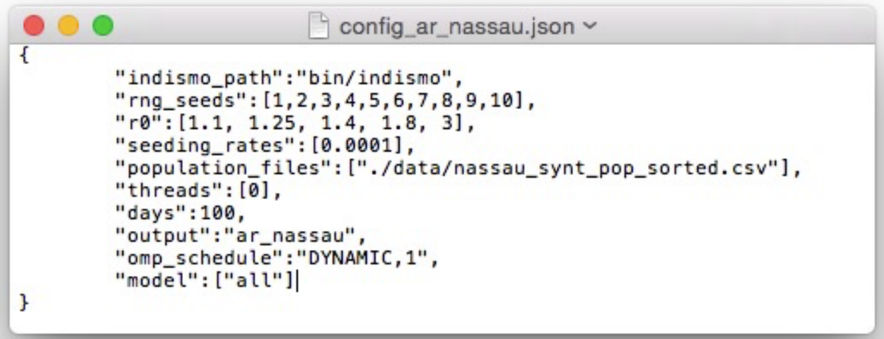
- `-o` or `--output_prefix`: Prefix for the output files, by default a time stamp.
- `-p` or `--population_file`: Population file.
- `-r` or `--r0`: Basic reproduction number: the number of secondary cases by a typical primary case in a complete susceptible population.
- `-n` or `--rng_seed`: Random number generator seed.
- `-s` or `--seeding_rate`: Epidemic seeding rate: fraction of initially infected people to start the epidemic.
- `-t` or `--transmission_rate`: Transmission rate: the probability that an infection is transmitted during a contact between two adults (+18 years) in the same social contact cluster.
- `-d` or `--days`: Number of days to simulate.

3.3 Sim Wrapper

A Python wrapper is provided to perform multiple runs with the C++ executable. The wrapper forwards the model configurations with command line arguments and merges the output. The wrapper is designed to be used with .json configuration files and examples are provided with the source code. For example:

```
./bin/sim_wrapper --config ./config/config_ar_nassau.json
```

will start the simulator with each configuration in the file illustrated in Figure 3.2. 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).



```
{
	"indismo_path":"bin/indismo",
	"rng_seeds":[1,2,3,4,5,6,7,8,9,10],
	"r0":[1.1, 1.25, 1.4, 1.8, 3],
	"seeding_rates":[0.0001],
	"population_files":["./data/nassau_synt_pop_sorted.csv"],
	"threads":[0],
	"days":100,
	"output":"ar_nassau",
	"omp_schedule":"DYNAMIC,1",
	"model":["all"]
}
```

Figure 3.2: Screen shot of a *sim_wrapper* configuration file.

Bibliography

[1] D. L. Chao, M. E. Halloran, V. J. Obenchain, and I. M. Longini Jr, "FluTE, a publicly available stochastic influenza epidemic simulation model," *PLoS Comp Biol*, vol. 6, no. 1, p. e1000656, 2010.

[2] J. J. Grefenstette, S. T. Brown, R. Rosenfeld, J. DePasse, N. T. Stone, P. C. Cooley, W. D. Wheaton, A. Fyshe, D. D. Galloway, A. Sriram, H. Guclu, T. Abraham, and D. S. Burke, "FRED (A Framework for Reconstructing Epidemic Dynamics): an open-source software system for modeling infectious diseases and control strategies using census-based populations," *BMC Public Health*, vol. 13, no. 1, p. 940, 2013.

[3] RTI International, "2010 RTI U.S. synthetic population ver. 1.0," *Downloaded from URL: http://www.epimodels.org/midas/pubsyntdata1.do*, 2014.

[4] W. Wheaton, "2010 U.S. synthetic population quick start guide. RTI international," *Downloaded from URL: http://www.epimodels.org/midasdocs/SynthPop/2010_synth_pop_ver1_quickstart.pdf*, 2014.