Hijacking Malaria Simulators with Probabilistic Programming

Bradley J. Gram-Hansen [*1] **Christian Schröder de Witt** [*1]
Tom Rainforth [2] **Philip H.S. Torr** [1] **Yee Whye Teh** [2] **Atılım Güneş Baydin** [1]

Abstract

Epidemiology simulations have become a fundamental tool in the fight against the epidemics of various infectious diseases like AIDS and malaria. However, the complicated and stochastic nature of these simulators can mean their output is difficult to interpret, which reduces their usefulness to policymakers. In this paper, we introduce an approach that allows one to treat a large class of population-based epidemiology simulators as probabilistic generative models. This is achieved by *hijacking* the internal random number generator calls, through the use of an universal probabilistic programming system (PPS). In contrast to other methods, our approach can be easily retrofitted to simulators written in popular industrial programming frameworks. We demonstrate that our method can be used for interpretable introspection and inference, thus shedding light on black-box simulators. This reinstates much needed trust between policymakers and evidence-based methods.

1. Introduction

Ending the epidemics of AIDS, tuberculosis, malaria and other infectious diseases by 2030 is a key target within the Good Health & Well-Being section of the UN Sustainable Development Goals (UN, 2017; 2018). However, despite decades of substantial international efforts, these diseases kill hundreds of million people a year. For example, malaria still annually kills about a quarter of a million children under the age of 5 in Africa alone.

To reach the WHO's target of reducing malaria incidence and mortality rates by at least 90% by 2030, policymakers are increasingly turning to evidence-based methods, thus oftentimes relying on computational simulations (WHO, 2015). These simulations allow policymakers to infer critical information on disease dynamics and make predictions about the impacts of policies before they are rolled out. This frequently increases the effectiveness of interventions and thus ultimately saves resources, or even lives. For example, it has been shown that mass vaccination may be largely ineffective in regions of large transmission rates, but may play a crucial role in areas of low transmission (Cameron et al., 2015).

Malaria epidemiology is governed by a complex set of drivers, few of which can be understood in isolation (Cameron et al., 2015; Autino et al.; Smith et al., 2008; Bershteyn et al., 2018). These include within-host dynamics, population-specific traits and even local geography. Comprehensive modeling of all of these components remains challenging, particularly in a region-specific context. Computational epidemiology simulators have to reflect these complexities and are usually stochastic in nature. This can make simulation output highly non-trivial to interpret, particularly when trying to draw desired inferences coupled with observed data (Mwendera et al.; Ferris et al.).

In this paper, we introduce a novel method that allows one to shed light on the inner workings of a large class of population-based stochastic simulators. We achieve this by extending the work of Baydin et al. (2018) by interpreting such population-based simulators as probabilistic generative models within the framework of universal probabilistic programming (UPP) (Le et al., 2017). To this end, we *hijack* existing simulators by overriding their internal random number generators. Specifically, by replacing the existing low-level random number generator in a simulator with a call to a purpose-built UPP "controller", which can thus control, track and manipulate the stochasticity of the simulator.

This allows for a variety of tasks to be performed on the hijacked simulator, such as running inference (by conditioning the values of certain draws and manipulating others), uncovering stochastic structure, and automatically producing result summaries, such as establishing the probability of different program paths/traces. By providing a common abstraction framework for different simulators, our approach further allows for easy and direct comparison between re-

[*]Equal contribution [1]Department of Engineering Science, University of Oxford, UK [2]Department of Statistics, University of Oxford, UK. Correspondence to: Bradley J. Gram-Hansen <bradley@robots.ox.ac.uk>.

Appearing at the International Conference on Machine Learning AI for Social Good Workshop, Long Beach, United States, 2019.

lated or competing simulators, a characteristic that is valuable in the context of epidemiology simulators (Ferris et al.). We provide a case study of the above in Section 4.

Our framework already supports application to simulators written in 13 general-purpose programming languages, and is easily extensible. This is crucial as, given the enormous code size and complexity, rewriting epidemiology simulators using a dedicated universal probabilistic programming language, such as Pyro (Bingham et al., 2019), is often infeasible.

In time, we hope our approach will play a critical role in bringing recent advancements in probabilistic programming to bear on the vast array of existing simulators used throughout the sciences, thereby providing wide-ranging impacts across a number of fields.

This paper first gives an overview of existing malaria simulators (Section 2), and proceed by introducing the necessary background on the *pyprob* framework and the concept of universal probabilistic programming (Section 3). Our approach is then demonstrated and analysed in the context of a malaria case study (Section 4).

2. Simulating Diseases

In-silico simulators have become a crucial tool in evidence-based decision-making within a large number of disciplines, including statistical physics (Landau & Binder, 2014), financial modeling (Jäckel, 2002), weather prediction (Evensen, 1994), epidemiology (Smith et al., 2008) and many others. In many cases, simulation output can augment or even replace real data that may otherwise be costly or even impossible to generate. Recent advances in hardware have enabled simulations to model increasingly complex systems. Epidemiology studies the prevalence and spreading of diseases across populations. Recent advances in hardware have enabled simulations to model the dynamics of infectious diseases, such as malaria, in ever greater detail.

2.1. Epidemiology Simulators

Two the most advanced malaria simulators, namely EMOD (Bershteyn et al., 2018) and OpenMalaria (Smith et al., 2008), have proven to be particularly valuable to policymakers. OpenMalaria is based on microsimulations of *Plasmodium falciparum* in humans and was originally developed to simulate the impacts of malaria vaccines within simple villages or districts. Compared with OpenMalaria, EMOD is able to simulate a variety of additional drivers, including complex geographies complete with migration and a large number of policy interventions. Both EMOD and OpenMalaria are open source and implemented in C++.

3. Hijacking Simulators

Probabilistic programming (Gordon et al., 2014; Staton et al., 2016; Kozen, 1979) can be used to express probabilistic models and consequently perform automated inference in these. Once a probabilistic model has been expressed in a probabilistic programming language, a wide range of inference techniques, such as Markov chain Monte Carlo (MCMC) (Geyer, 1992), black-box variational inference (VI) (Ranganath et al., 2014) and amortized inference(Le et al., 2016), can be used by non-experts in an automated fashion.

Hijacking a simulator describes the process by which a simulator's random number generators are replaced by calls to external sampling procedures, which are controlled by a probabilistic programming system (PPS). In practice, this amounts to performing a small number of surgical incisions into the simulator's source code in order to replace built-in calls to random number generators. E.g., given a simulator written in C++/Boost (Schäling, 2011), a Gaussian distribution object *boost::normal* is replaced with the corresponding *pyprob_cpp* distribution object, namely *pyprob_cpp::distributions::Normal*. Sampling from this distribution is then done by requesting the PPS to send a sample back to the simulator.

The PPS and the simulator exchange xTensor objects[1] through TCP or IPC using a generic FlatBuffers[2] protocol. On the PPS side, sampling is done using the deep learning framework PyTorch (Paszke et al., 2017).

After a variable has been sampled, using *pyprob_cpp::sample*, it is sent by the PPS to the simulator, at the same recording the simulator execution trace as a side effect. This allows the PPS to construct sample trace probabilities and other summary statistics (cf. Section 4).

Finally, the entry point to the simulator, i.e., *main* in C++, is replaced by a special *forward* call, in this case *pyprob_cpp::forward*. This allows the PPS to generate rollouts from the simulator remotely. A pictorial overview of this process is depicted in Figure 3.

Population-based simulators create a trace per population member, in contrast to event-based simulators that create only a single trace per *forward* call (Baydin et al., 2018). This means that, e.g., in OpenMalaria, a standard scenario simulation rollout over the span of 3 years with a population of size $n = 5000$ (Smith et al., 2008) will generate about *four terabytes* of raw trace data. Unlike the simulator used

[1] https://xtensor.readthedocs.io/en/latest/

[2] http://google.github.io/flatbuffers/

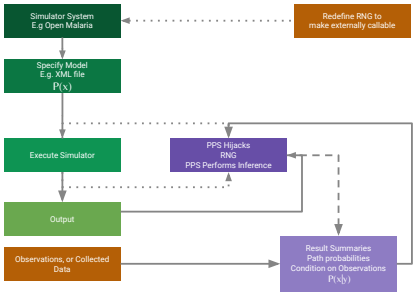

Figure 1. This flow-chart provides an overview of the process of how our we hijack a generic population-based epidemiology simulator, such as OpenMalaria, and how we modify the simulator to to hijack the random number generator (RNG). It demonstrates how information is exchanged between the simulator and the PPS. Green represents events linked to the simulator, purple corresponds to events occurring in the PPS and brown represents an external processes.

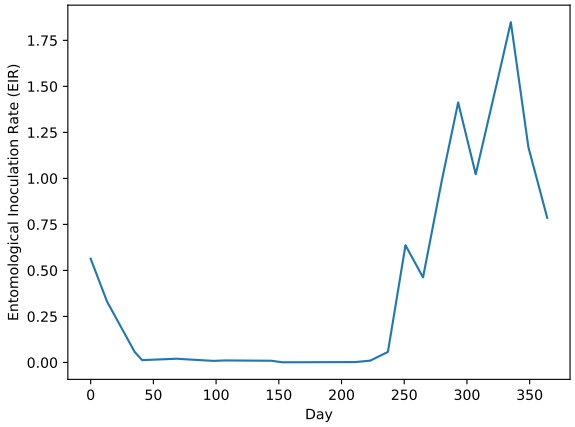

Figure 2. Seasonal Entomological Inoculation Rate (EIR) for the Ifakara scenario. Data is averaged over 30 day periods.

by Baydin et al. (2018), this amount of data cannot feasibly be kept in RAM, which makes *a posteriori* trace analysis very inefficient. In order to deal with the shortcomings of *pyprob* in a population-based simulator context, we therefore extend the framework to be able to do trace analysis on the fly.

By extending *pyprob* to handle population-based simulators, the PPS can now track all stochastic random variables that are created within the simulator, which then allows us to generate trace plots and path probabilities associated to the execution paths of the program. The corresponding increase of simulator transparency helps reinstate much needed trust between policymakers and evidence-based methods.

4. Case Study: Ifakara, Tanzania

Ensemble methods are commonly used in statistics in order to combine the predictive power of multiple models (Cameron et al., 2015; Smith et al.). To this end, recent work has attempted to characterise the similarities and differences between two of the most advanced malaria epidemiology simulators, EMOD (Bershteyn et al., 2018) and OpenMalaria (Smith et al., 2008). Evaluation is usually done by comparing a number of output parameters across a range of hand-crafted standard scenarios reflecting different geographical locations across Africa (Smith et al.).

In the following, we illustrate how our method introduces a novel introspection paradigm. By extracting trace graphs from population-based simulators, policymakers can ask specific questions about properties of the model trace flow and not just the outcomes of the model, thus providing additional interpretability to the decision-making process.

To illustrate the above, we present simulation output generated from a scenario resembling local conditions in the town of Ifakara, Tanzania. Both EMOD and OpenMalaria are configured to simulate a single population node of $n = 100$ and assume constant climatic conditions and no migration over the simulation period of three years. Please refer to Figure 2 for the seasonal Entomological Inoculation Rate (EIR), a measure of infectivity.

We provide examples of the generated trace plots from the connection between the simulator and the PPS in Figure 3.

We can see from the addressing schemes $A_1, \ldots, A_N$ (Tables 2 and 4) what physical events are connected to each other and how outputs in EMOD are generated from a different set of procedures as compared to OpenMalaria. By having access to such diagrams, users can internally evaluate and scrutinize the decisions that the simulator is making.

This is important for policymakers, or general non-experts, as it not only details how we arrive at the given outputs, but it provides an understanding of which processes were most crucial in determining those outputs as can be seen from the path probabilities assigned to each of the vertices.

Additionally, by associating nodes in trace graphs representing the same physical processes within different models, the significance of model detail can be evaluated. For example, the full trace presented in Table 4 represent a sampling step associated with within-host dynamics of the malaria parasite *Falciparum* in OpenMalaria node $A1$. The same physical process also occurs in EMOD's trace graph at position $A7$ (see Section 2). Comparisons like these could help developers better control model complexity, and even provide an alternative testing and debugging paradigm.

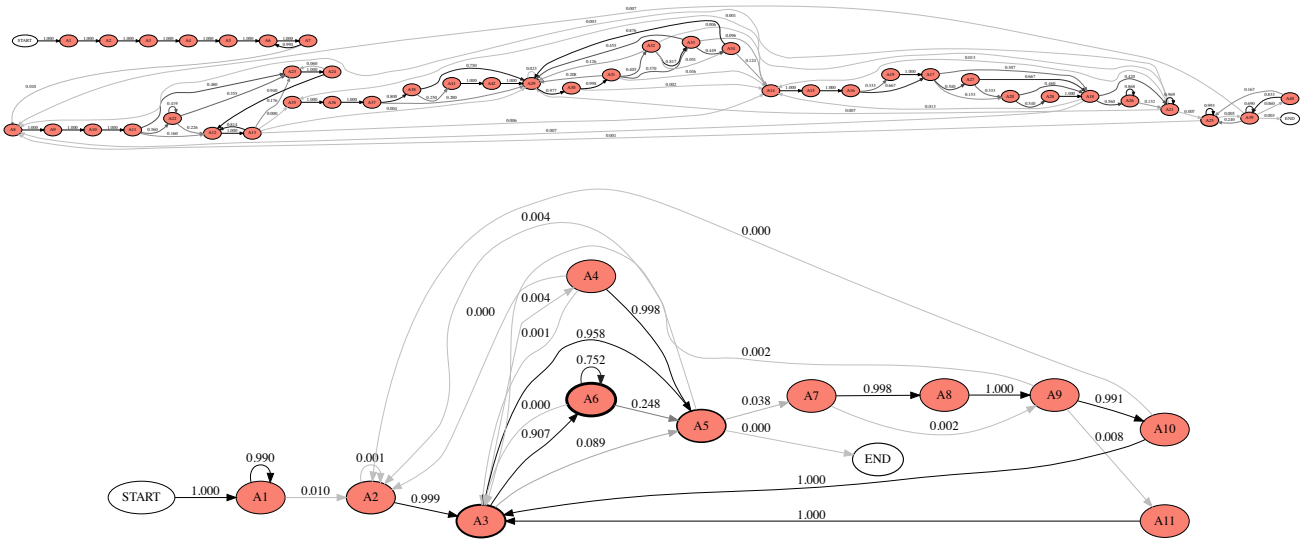

Figure 3. Here we run two equivalent models, compare the corresponding trace paths and corresponding path probabilities taken by the thousands of random variables generated internally within the simulators. ***Top:*** The specified model run in EMOD. ***Bottom:*** The specified model in OpenMalaria.

Table 1. An example of an address generated for the model run in the OpenMalaria simulator. We can see that A1 relates to Generating a member of the human population who may or may not be infect with the malaria disease. We get something similar for EMOD, except this relates to A7 in the EMOD program execution.

Address ID	Full address

A1	[forward()+0x204; OM::Simulator::
	start(scnXml::Monitoring const)+0x28a;
	OM::Population::createInitialHumans()+0x94;
	OM::Population::newHuman(OM::SimTime)+0x5c;
	OM::Host::Human::Human(OM::SimTime)+0x12b;
	OM::WithinHost::WHInterface::
	createWithinHostModel(double)+0x99;
	OM::WithinHost::DescriptiveWithinHostModel
	::DescriptiveWithinHostModel(double)+0x3a;
	OM::WithinHost::WHFalciparum::
	WHFalciparum(double)+0xe6;
	OM::util::random::
	gauss(double, double)+0xb4]__Normal

⋮	⋮

Table 2. An interpretation table for each of the address of the overall trace generated from the corresponding OpenMalaria model.

Address ID	Interpretation
A1	Generate a human in the population within host dynamics
A2	Generate another human in the population within host dynamics
A3	The population is updated and a new human, or humans, may get infected
A4, A5	Potential child deaths within the population are simulated
A6	Determines parasite density of an individual infection
A7	Models how the disease is progressing within the infected humans
A8	Models how the disease is progressing within the population
A9	Models how the disease is progressing within the infected humans after the population has been updated
A10	Full clinical update on the population for those without severe or no Malaria infection.
A11	Full clinical update on the population for those with severe Malaria infections

5. Discussions and Future Work

In this work we have demonstrated a method that enables one to hijack population-based simulators, extending the work of Baydin et al. (2018). We applied our method to two malaria-orientated population-based simulators and generated a variety of trace graphs. Finally, we have shown how our system enables policy makers and non-experts to analyse simulator outputs in a way previously unavailable in the field of epidemiology.

To extend our work further we aim to implement additional tools that will facilitate complicated inference procedures that condition on simulator output. We will also evaluate additional scenarios across Africa and Southeast Asia to better understand the similarities and differences between EMOD and OpenMalaria.

6. Acknowledgments

We thank Ewan Cameron of MAP at the Big Data Institute and the OpenMalaria team at the Swiss Tropical Health Institute for their time and help. BGH is supported by the EPRSC Autonomous Intelligent Machines and Systems grant. CSW is supported by the project Free the Drones (FreeD) under the Innovation Fund Denmark and Microsoft. YWT's and TR's research leading to these results has received funding from the European Research Council under the European Union's Seventh Framework Programme (FP7/2007-2013) ERC grant agreement no. 617071. AGB and PH are supported by EPSRC/MURI grant EP/N019474/1 and AGB is also supported by Lawrence Berkeley National Lab.

References

Autino, B., Noris, A., Russo, R., and Castelli, F. Epidemiology of malaria in endemic areas. 4 (1). ISSN 2035-3006. doi: 10.4084/MJHID.2012.060. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3499992/.

Baydin, A. G., Heinrich, L., Bhimji, W., Gram-Hansen, B., Louppe, G., Shao, L., Cranmer, K., Wood, F., et al. Efficient probabilistic inference in the quest for physics beyond the standard model. *arXiv preprint arXiv:1807.07706*, 2018.

Bershteyn, A., Gerardin, J., Bridenbecker, D., Lorton, C. W., Bloedow, J., Baker, R. S., Chabot-Couture, G., Chen, Y., Fischle, T., Frey, K., et al. Implementation and applications of emod, an individual-based multi-disease modeling platform. *Pathogens and disease*, 76(5):fty059, 2018.

Bingham, E., Chen, J. P., Jankowiak, M., Obermeyer, F., Pradhan, N., Karaletsos, T., Singh, R., Szerlip, P., Horsfall, P., and Goodman, N. D. Pyro: Deep universal probabilistic programming. *The Journal of Machine Learning Research*, 20(1):973–978, 2019.

Cameron, E., Battle, K. E., Bhatt, S., Weiss, D. J., Bisanzio, D., Mappin, B., Dalrymple, U., Hay, S. I., Smith, D. L., Griffin, J. T., et al. Defining the relationship between infection prevalence and clinical incidence of plasmodium falciparum malaria. *Nature communications*, 6:8170, 2015.

Evensen, G. Sequential data assimilation with a nonlinear quasi-geostrophic model using monte carlo methods to forecast error statistics. *Journal of Geophysical Research: Oceans*, 99(C5):10143–10162, 1994.

Ferris, C., Raybaud, B., and Madey, G. OpenMalaria and EMOD: A case study on model alignment. In *Proceedings of the Conference on Summer Computer Simulation*, SummerSim '15, pp. 1–9. Society for Computer Simulation International. ISBN 978-1-5108-1059-4. URL http://dl.acm.org/citation.cfm?id=2874916.2874952. event-place: Chicago, Illinois.

Geyer, C. J. Practical markov chain monte carlo. *Statistical science*, pp. 473–483, 1992.

Gordon, A. D., Henzinger, T. A., Nori, A. V., and Rajamani, S. K. Probabilistic programming. In *Proceedings of the Future of Software Engineering*, pp. 167–181. ACM, 2014.

Jäckel, P. *Monte Carlo Methods in Finance*. The Wiley Finance Series. Wiley, 2002. ISBN 9780471497417. URL https://books.google.co.uk/books?id=jG6BQgAACAAJ.

Kozen, D. Semantics of probabilistic programs. In *Foundations of Computer Science, 1979., 20th Annual Symposium on*, pp. 101–114. IEEE, 1979.

Landau, D. P. and Binder, K. *A Guide to Monte Carlo Simulations in Statistical Physics*. Cambridge University Press, 4 edition, 2014. doi: 10.1017/CBO9781139696463.

Le, T. A., Baydin, A. G., and Wood, F. Inference compilation and universal probabilistic programming. *arXiv preprint arXiv:1610.09900*, 2016.

Le, T. A., Baydin, A. G., and Wood, F. Inference compilation and universal probabilistic programming. In *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1338–1348, Fort Lauderdale, FL, USA, 2017. PMLR.

Mwendera, C. A., de Jager, C., Longwe, H., Kumwenda, S., Hongoro, C., Phiri, K., and Mutero, C. M. Challenges to the implementation of malaria policies in malawi. 19(1):194. ISSN 1472-6963. doi: 10.1186/s12913-019-4032-2. URL https://doi.org/10.1186/s12913-019-4032-2.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

Ranganath, R., Gerrish, S., and Blei, D. Black box variational inference. In *Artificial Intelligence and Statistics*, pp. 814–822, 2014.

Schäling, B. *The boost C++ libraries*. Boris Schäling, 2011.

Smith, T., Ross, A., Maire, N., Chitnis, N., Studer, A., Hardy, D., Brooks, A., Penny, M., and Tanner, M. Ensemble modeling of the likely public health impact of a pre-erythrocytic malaria vaccine. 9(1):e1001157. ISSN 1549-1676. doi: 10.1371/journal.pmed.1001157. URL https://journals.plos.org/plosmedicine/article?id=10.1371/journal.pmed.1001157.

Smith, T., Maire, N., Ross, A., Penny, M., Chitnis, N., Schapira, A., Studer, A., Genton, B., Lengeler, C., Tediosi, F., et al. Towards a comprehensive simulation model of malaria epidemiology and control. *Parasitology*, 135(13):1507–1516, 2008.

Staton, S., Yang, H., Wood, F., Heunen, C., and Kammar, O. Semantics for probabilistic programming: higher-order functions, continuous distributions, and soft constraints. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pp. 525–534. ACM, 2016.

UN. The sustainable development goals report 2018. 2018. URL https://doi.org/10.18356/7d014b41-en.

UN, U. N. The 2030 agenda for sustainable development. 2017. URL https://www.refworld.org/docid/57b6e3e44.html.

WHO. Global technical strategy for malaria 2016-2030, 2015. URL http://www.who.int/malaria/areas/global_technical_strategy/en/.