# ELFI: Engine for Likelihood-Free Inference

**Antti Kangasrääsiö**[1], **Jarno Lintusaari**[1], **Kusti Skytén**[1], **Marko Järvenpää**[1], **Henri Vuollekoski**[1], **Michael Gutmann**[2], **Aki Vehtari**[1,5], **Jukka Corander**[3,4,5], **Samuel Kaski**[1,5]

[1]Helsinki Institute for Information Technology HIIT, Department of Computer Science, Aalto University, Helsinki, Finland
[2]School of Informatics, University of Edinburgh, Edinburgh, UK
[3]Helsinki Institute for Information Technology HIIT, Department of Mathematics and Statistics, University of Helsinki, Helsinki, Finland
[4]Department of Biostatistics, University of Oslo, Norway
[5]Equal contributions

## Abstract

We introduce an *Engine for Likelihood-Free Inference* (ELFI), a software package for approximate Bayesian inference that can be used when the likelihood function is difficult to evaluate or unknown, but a generative simulator model exists. The software is in Python, and its modular library design emphasizes both ease-of-use and expandability, allowing arbitrary user-defined simulators and implementation of new inference methods with minimal effort. Probabilistic inference models can be represented intuitively as graphs, and users can execute the inference in a computational environment best suited for their needs, from single laptops to cluster computers. The whole inference pipeline is automatically parallelized, and intermediate results may be stored to disk for later use. The package includes implementations of some of the most advanced likelihood-free inference techniques. One example of these is BOLFI, which estimates the discrepancy function using Gaussian processes and uses Bayesian optimization for parameter search, which has recently been shown to accelerate likelihood-free inference up to several orders of magnitude.

## 1 Introduction

The likelihood function associated with a probabilistic model is often difficult or impossible to evaluate directly, in particular when the generative model is only defined as an executable simulator. Inference of such models may be accomplished with indirect inference, synthetic likelihood or Approximate Bayesian Computation (ABC) [2, 7, 9]. All of these methods are based on the assumption that the qualities of the simulated data vary relatively smoothly with changes in the model parameters. The general approach is that observed data are systematically compared to data acquired from simulations with sampled parameters. For example, in the case of ABC, this allows us to create an approximate posterior distribution for the parameters of interest. [5, 6, 4]

Multiple software packages for ABC already exist in many programming languages [8]. However, with the increasing popularity of the Python programming language also among data scientists, an expandable Python package for likelihood-free inference is attractive, in particular because the most efficient machine learning based tools for accelerating ABC inference have recently become available in Python and these options remain unavailable in existing ABC packages. We introduce a general, modular ABC framework: Engine for Likelihood-Free Inference (ELFI).

30th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

## 2 Description of the software package

The ELFI Python package has been designed with three main requirements in mind. First ELFI is easy to use and features a user interface that allows the researcher to define the inference problem in the form of graphical models. This supports the intuitive creation of complex probabilistic models. In likelihood free inference (LFI), the probabilistic model contains parameters for which the likelihood is defined implicitly by a simulator model. The simulator may be any user-defined Python function or a binary executable that fulfils minimal interface requirements.

Second, any modern software should take advantage of the increasing number of processing units in computing hardware. Therefore, ELFI has been implemented so that running the user-defined simulator, the calculation of the similarity metric and feasible parts of the ABC algorithms, are automatically parallelized for distributed computing. The parallelization has been implemented with the Python package *Dask* [1], which scales well from a laptop computer up to a distributed cluster environment.

Third, ELFI is modular and follows the object-oriented paradigm. The development and testing of new inference methods on top of the framework requires minimal effort and provides all the advantages of existing functionality, such as parallelization, handling the states of the pseudo random number generators and persisting results to disk. Due to the object oriented approach, one can readily extend existing functionality. Currently included LFI methods are the rejection sampling, the sequential Monte Carlo sampling and the Bayesian Optimization for Likelihood-Free Inference (BOLFI) framework, which has been shown to reduce the total computational burden by several orders of magnitude [3].

ELFI is open-source under the liberal BSD3 license. This allows community effort in maintaining and further developing the library. While ELFI will remain in an active development by the authors, contributions are welcome and ELFI will give explicit credit to contributors.

More information including source code, documentation and examples can be found at the ELFI website: `http://elfi.readthedocs.io`.

## 3 Example use case: Moving average model

As an example use case for ELFI, consider the MA(2) model [5]. The zero-mean MA(2) model is a moving average of three consecutive independent and identically distributed components of white noise $(w_k)_{k \in Z} \sim N(0, 1)$ such that

$$x_t = w_t + \theta_1 w_{t-1} + \theta_2 w_{t-2} \tag{1}$$

with parameters $\theta_1, \theta_2 \in R$. Given observations $Y$, inference of unknown parameters $\theta_1, \theta_2$ is intractable due to the high stochasticity and difficult likelihood. In ELFI one could infer the parameters by defining a graph as in Figure 1: setting priors for $\theta_1$ and $\theta_2$, defining a simulator (MA2) that depends on those parameters, defining summary statistics $S1$ and $S2$ (e.g. autocovariances) that

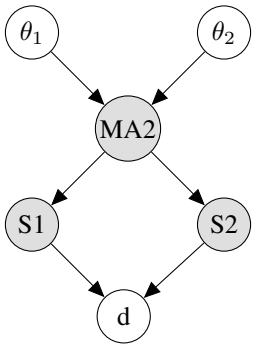


Figure 1: An example of a probabilistic model that ELFI uses to infer the parameters $\theta_1$ and $\theta_2$ in the MA(2) model.

depend on the simulated (and observed) data and finally a distance metric $d$ that depends on the summary statistics. The user chooses an inference algorithm, and the computations are automatically parallelized. Given the simulator and the required summary statistics and discrepancy functions, the ELFI part of the inference can be done in few lines of legible Python code:

```
import elfi

# the simulator, summary statistics and discrepancy functions
from X import MA2, acov1, acov2, L2dist

# Model definition as nodes: name, operation and parents
t1 = elfi.Prior('t1', 'uniform', 0, 2)
t2 = elfi.Prior('t2', 'uniform', 0, 2)
Y = elfi.Simulator('MA2', MA2, t1, t2, observed=y_obs)
S1 = elfi.Summary('S1', acov1, Y)
S2 = elfi.Summary('S2', acov2, Y)
d = elfi.Discrepancy('d', L2dist, S1, S2)

# Inference algorithm (Quantile-based rejection sampling)
rej = elfi.Rejection(d, [t1, t2])
n_samples = 10000
results = rej.sample(n_samples, quantile=0.01)

# More advanced inference algorithm (BOLFI) using Gaussian process
gp_model = elfi.GPyModel(input_dim=2, bounds=((0,2), (0,2)))
bolfi = elfi.BOLFI(d, [t1, t2], n_surrogate_samples=150,
                   model=gp_model)
posterior = bolfi.infer()
```

Note that using just uniform priors renders the MA(2) case unidentifiable [5]. The graphical model syntax of ELFI supports also hierarchical priors, assuming the user has defined their distributions:

```
from X import Prior_t1, Prior_t2

# replace the definitions for priors in the above code
t1 = elfi.Prior('t1', Prior_t1, 2)
t2 = elfi.Prior('t2', Prior_t2, t1, 1) # t2 depends on t1
```

**Acknowledgments**

This work has been partly supported by the Academy of Finland (Finnish Centre of Excellence in Computational Inference Research COIN).

## References

[1] Dask Development Team (2016). *Dask: Library for dynamic task scheduling*.

[2] Gourieroux, C., Monfort, A., and Renault, E. (1993). Indirect inference. *Journal of Applied Econometrics*, 8(S1):S85–S118.

[3] Gutmann, M. U. and Corander, J. (2016). Bayesian optimization for likelihood-free inference of simulator-based statistical models. *Journal of Machine Learning Research*, 17(125):1–47.

[4] Lintusaari, J., Gutmann, M. U., Dutta, R., Kaski, S., and Corander, J. (2016). Fundamentals and recent developments in approximate Bayesian computation. *Systematic Biology*, in press.

[5] Marin, J.-M., Pudlo, P., Robert, C. P., and Ryder, R. J. (2012). Approximate Bayesian computational methods. *Statistics and Computing*, 22(6):1167–1180.

[6] Sunnåker, M., Busetto, A. G., Numminen, E., Corander, J., Foll, M., and Dessimoz, C. (2013). Approximate Bayesian computation. *PLoS Computational Biology*, 9(1):e1002803.

[7] Tavaré, S., Balding, D. J., Griffiths, R. C., and Donnelly, P. (1997). Inferring coalescence times from dna sequence data. *Genetics*, 145(2):505–518.

[8] Wikipedia (2016). Approximate Bayesian computation — Wikipedia, the free encyclopedia. [Online; accessed 7-Oct-2016].

[9] Wood, S. N. (2010). Statistical inference for noisy nonlinear ecological dynamic systems. *Nature*, 466(7310):1102–1104.